SUBSISTEMA DE FACHADAS DE INTERPRETACIÓN DE PERFILES PARA EL MÓDULO DE PERSONALIZACIÓN DE CONTENIDO EN EL MARCO DEL PROYECTO EUROPEO EU4ALL

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Proyecto Fin de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Bernardo Lezcano Pastor

Tutor: Ángel García Crespo

Índice General

1 Introducción

EU4ALL es un proyecto europeo que pretende crear un marco de trabajo, basado en estándares, que facilite la integración de las metodologías con un amplio rango de sistemas de aprendizaje electrónico (*e-learning*), de ahí el significado de sus siglas, Europa Unida Para la Asistencia de Aprendizaje Permanente (*Europe United Approach For Accesible Lifelong Learning*).

El aprendizaje es necesario dentro de una economía basada en el conocimiento, la educación y el trabajo y dentro de este aprendizaje, las tecnologías de la información juegan un papel muy importante en la actualidad.

Es necesario que esta información pueda llegar a cualquier persona independientemente de las discapacidades que pueda tener y si la tecnología utilizada es inadecuada o insuficiente para proporcionar la información a las personas discapacitadas, éstas se verán excluidas de la interconexión entre la educación y trabajo.

El proyecto europeo EU4ALL avanza en este concepto "ALL" (*Accesible Lifelong Learning*) creando una tecnología para todos, ayudando a saltar la barreras que puedan existir para personas con discapacidades.

Uno de los principales objetivos de cualquier sistema de gestión del aprendizaje es ser capaz de adaptarse dinámicamente a las necesidades del usuario. Esto se consigue con la Personalización de Contenido, de manera que los contenidos se transforman para que diferentes usuarios de diferentes niveles de discapacidad accedan a la información en un contexto adecuado.

El módulo de Personalización de Contenido es un núcleo interno del proyecto EU4ALL que facilitará un medio para entregar contenido a los usuarios, adaptado a sus necesidades y preferencias individuales que componen su perfil.

El Subsistema de Fachadas es una pequeña pieza que forma parte del módulo de personalización de contenido, dentro del proyecto EU4ALL, que proporciona un mecanismo para interpretar diferentes perfiles independientemente del estándar de modelado que se esté utilizando.

En este documento vamos a dar una visión general del proyecto EU4ALL viendo sus necesidades, objetivos y elementos que lo componen.

Daremos una vuelta por el módulo de personalización de contenido para conocer sus características, funcionamiento y ubicar el subsistema de fachadas dentro del proyecto EU4ALL.

Y como objetivo principal de este proyecto fin de carrera vamos crear un módulo de software que represente el subsistema de fachadas dentro del módulo de

personalización de contenido y que permita gestionar cualquier estándar de representación de perfiles de usuario.

Mostraremos la parte técnica del subsistema de fachadas con todo detalle mediante su diseño técnico e implementación y un manual de usuario.

2 Estructura del documento

En el capítulo 3 se describirá la situación en la que se encuentra el estudio y desarrollo de normas y estándares de accesibilidad del contenido de la web, así como las organizaciones que participan hasta llegar al momento donde se ve la necesidad de desarrollar este proyecto para crear un subsistema de fachadas, capaz de tratar de forma genérica los distintos estándares de modelado de perfiles de usuario, debido a la falta de existencia de un estándar común.

El capítulo 4 mostrará los objetivos que se pretenden cumplir con este proyecto, centrados en las necesidades con las que surge el desarrollo del módulo de software del subsistema de fachadas.

En el capítulo 5 veremos el efecto del proyecto orientado al impacto que el subsistema de fachadas dentro de la personalización de contenido tiene sobre los usuarios.

El capitulo 6 permitirá conocer el proyecto europeo EU4ALL del que forma parte este proyecto fin de carrera. Se describirán las necesidades por las que surge el proyecto EU4ALL, los objetivos que pretenden lograr y los resultados que se pueden conseguir, además los logros en accesibilidad de cualquier persona a la educación, también desde el punto de vista de la formación e investigación y desarrollo tecnológico.

Se describirán las características de la Personalización de Contenido y el módulo de software desarrollado dentro de EU4ALL para realizar esta tarea. Con la personalización se transforma el contenido para adaptarlo a las necesidades y preferencias del usuario. Es dentro de éste módulo donde se encuentra ubicado el Subsistema de Fachadas para la Interpretación de Perfiles.

En el capítulo 7 se pretende mostrar la utilidad del subsistema de fachadas dentro del módulo de personalización de contenido. Se describen los modelos de datos incluidos en la Personalización de Contenido que se ven afectados en la interpretación de perfiles y el funcionamiento del proceso de personalización ubicando el subsistema de fachadas en los distintos puntos del proceso en los que va a ser útil.

El capitulo 8 mostrará con todo detalle la parte técnica del proyecto desde los requisitos funcionales hasta su implementación. Describirá los distintos escenarios surgidos para el comportamiento que se quiere que desempeñe el sistema, la solución técnica en la que se apoya el diseño y un conjunto de diagramas que junto con una descripción detallada de los mismos permitirá comprender el funcionamiento del subsistema.

El capitulo 9 describiremos las conclusiones obtenidas tanto por la realización de este trabajo como personales y una línea futura de desarrollo.

Para una mejor comprensión a la hora de utilizar el subsistema, en el capitulo 10 crearemos un manual de usuario que muestre los datos de configuración necesarios para su funcionamiento y un ejemplo en código de cómo utilizarlo.

En el anexo mostraremos la planificación y describiremos el presupuesto necesario para llevar a cabo el proyecto.

3 Estado del arte

Hay una amplia gama de necesidades especiales, desde movilidad y psíquicas, a incapacidad sensorial (vista, oído, tacto), discapacidad física (movilidad y destreza), y dificultades cognitivas y de aprendizaje (incluida la palabra, problemas de expresión, dislexia y otros problemas de aprendizaje específicos).

La discapacidad afecta de forma diferente a las personas que aprenden la interacción con cualquier tecnología de aprendizaje, variando ampliamente según las diferentes estrategias individuales de afrontarlas, las tecnologías de asistencia empleadas y las preferencias personales.

Los estudiantes con necesidades especiales han variado, teniendo perfiles individuales de aprendizaje y estilos, intereses y objetivos. Los estudiantes con discapacidades pueden necesitar medios particulares para acceder a la enseñanza y el aprendizaje, requiriendo que el contenido y las actividades de aprendizaje se adapten a sus preferencias de interacción con el ordenador.

Actualmente las pautas sobre accesibilidad vienen marcadas por la WAI, (*Web Accessibility Initiative*) Iniciativa de Accesibilidad a la Web creada en 1988 bajo el marco de la W3C, *World Wide Web Consortium*, organización internacional que orienta y estructura el desarrollo global de la WWW.

La WAI, en coordinación con otras organizaciones, promueve la accesibilidad a la Web a través de distintos grupos de trabajo.

En 1999 publicó las Pautas de Accesibilidad del Contenido en la Web, una recomendación de la W3C para el diseño de sitios Web accesibles, que es la referencia obligatoria en éste área a nivel internacional (www.w3.org/WAI).

Existen 3 conjuntos de directrices de la W3C WAI que desempeñan un papel importante en el logro de sitios web accesibles.

1. *Web Content Accessibility Guidelines WCAG (Version 1.0 and 2.0)*. Normas de accesibilidad de contenidos Web.

2. *Web Authoring Tools Accessibility Guidelines (ATAG 1.0 and 2.0)*. Normas de accesibilidad de herramientas Web de autor.

3. *User Agents Accessibility Guidelines (UAAG 1.0)*. Normas de accesibilidad de agentes de usuario.

Desde el punto de vista de accesibilidad, podemos tener diferentes enfoques:

- Diseño de objetos accesibles universalmente.

- Personalización de los recursos, seleccionándolos y adaptándolos para satisfacer al usuario.

Además existe un enfoque de estudio que son los asistentes personalizados, que deben saber acerca de los usuarios y aprender sobre sus preferencias y procedimientos observando el acceso de los usuarios a los servicios desplegados.

Éste modelo se clasifica en:

- Información de cómo interactúan los usuarios con los servicios.
- Información sobre los servicios que ha usado el usuario.
- Explicación del resultado de determinadas acciones sobre los servicios.

El modelado detallado de usuario guarda toda la información referente a los usuarios, lo que puede permitir a un sistema avanzar en su grado de accesibilidad, consiguiendo adaptarlo a las características concretas de cada usuario [2].

Existen diferentes especificaciones y estándares para la representación de características de usuarios:

- *IMS LIP + IMS AccLIP*
- *CC/PP [11]*
- *ISO IEC JTC1 SC36 Individualised Adaptability and Accessibility for Learning Education and Training*

Además un conjunto de normas que se utilizan para los modelos de usuario y descripción de contenido.

- *Modelo SCORM*
- *IMS-CP*
- *IMS-MD/LOM*
- *IMS QTI*
- *CAC-LIP*
- *CAC-MD*

Ya que nos encontramos dentro del campo del *e-learning* que es relativamente joven e inmaduro, estamos en una fase previa de definición de un estándar suficientemente completo. En esta fase es normal una confusión que responde a la elaboración descoordinada de especificaciones por parte de diferentes organizaciones. Este trabajo en paralelo desemboca en un escenario confuso inundado por las siglas de cada una de las propuestas desarrolladas. *IMS IEEE, ADL/SCORM, AICC, MIT/OCW/OKI* son solo unos ejemplos de la gran cantidad de consorcios, iniciativas, organismos e instituciones

implicados en el proceso de estandarización del *e-learning*. Esta diversidad, originada por la poca madurez del dominio, aumenta el riesgo de elegir el estándar perdedor. Invertir mucho esfuerzo en el desarrollo o aprendizaje de un estándar que finalmente no tendrá aceptación comercial puede tener un alto coste.

Afortunadamente existe una tendencia unificadora e integradora entre los principales desarrolladores de especificaciones agrupadas bajo las iniciativas de *IMS [7]*, *IEEE*, *LTSC*, e *ISO/IEC* que puede ayudar a simplificar y generalizar la creación y adopción de un único estándar educativo que sirva como referencia.

Tanto los datos de perfiles de usuario como los metadatos de los recursos se definen en la especificación *IMS AccessForAll*. Estas especificaciones están siendo aprobadas con una *ISO* bajo nivel de desarrollo que conocemos como *SC34*. También se están perfilando para la *IEEE LOM* y la *Dublín Core Metadata*.

Hasta el momento no existe un estándar único de modelo de usuario. La representación de un perfil de usuario puede ser por lo tanto representada por cualquiera de los estándares que se existen actualmente. En este punto es donde cobra toda su importancia el subsistema de fachadas, ya que proporcionar el mecanismo que va a permitir al sistema trabajar con los datos del usuario sin preocuparse del estándar de perfil que se esté utilizando en la representación de los mismos.

4 Objetivos

El Objetivo con el que surge este proyecto fin de carrera es desarrollar un módulo de software dentro del proyecto EU4ALL para cubrir una de sus necesidades como es, la interpretación de perfiles de usuario dentro del módulo de personalización de contenido.

Este módulo de software es lo que llamamos el subsistema de fachadas y los objetivos que queremos alcanzar con su desarrollo son:

- Proporcionar un mecanismo que facilite al módulo de personalización de contenido la obtención de las características del usuario que accede al sistema con cualquier estándar de definición de perfiles.

- Abstraer la representación interna que tendrá el sistema sobre el usuario más allá de las especificaciones y estándares que existen actualmente para tratar perfiles de usuario. El usuario únicamente será consciente de que el sistema ha actuado en base a las especificaciones de su perfil sin preocuparse del estándar utilizado.

- Crear una arquitectura que permita aislar el tratamiento interno del perfil de usuario con respecto a la implementación concreta de ese perfil. Esto nos permitirá poder añadir nuevas implementaciones de estándares de perfiles de usuario de manera sencilla, sin necesidad de modificar la arquitectura.

Más allá del objetivo inicial, pretendemos que el diseño y desarrollo de este Subsistema de Fachadas constituya un marco de desarrollo escalable y con alto grado de reusabilidad tanto dentro del proyecto EU4ALL como en otros sistemas.

Escalable, para que permita de manera sencilla cambios sobre su diseño.

Reusable, para que tanto otros módulos dentro del proyecto EU4ALL como otros sistemas puedan utilizarlo con un grado mínimo de adaptación.

En este documento, queremos proporcionar una visión general de la funcionalidad del Módulo de Software de Personalización de Contenido (CPSM) y ver con cierto detalle las medidas requeridas para mostrar a los usuarios los recursos más adecuados.

Vamos a mostrar una implementación de referencia para comprobar la viabilidad del subsistema diseñado y un manual de usuario que permita comprender mejor el diseño.

Mostraremos el diseño técnico y la implementación del subsistema de fachadas con la adaptación del mismo a la interpretación de modelado de perfiles de usuario.

5 Ámbito

Se puede decir que como EU4ALL trata de definir una arquitectura abierta de servicios (SOA) para plataformas de *e-learning*, el módulo de personalización de contenido puede tener un gran impacto en las comunidades educativas a la hora de incrementar la satisfacción del usuario con su uso.

Tendríamos dos tipos de usuarios que se beneficiarán de las soluciones aportadas por el proyecto:

a) Usuarios de sistemas, tales como arquitectos, desarrolladores e integradores de sistemas que conciben y desarrollan aplicaciones de aprendizaje con tecnología avanzada y los proveedores de servicios de apoyo y contenido, como los autores de contenidos de aprendizaje electrónico y materias, apoyo pedagógico, profesores.

b) Usuarios finales, estudiantes con necesidades especiales como las personas con problemas de movilidad, visuales, auditivos y de luminosidad.

El desarrollo realizado en este proyecto es una clave fundamental a la hora de aislar los estándares utilizados del comportamiento del Módulo de PC (Personalización de Contenido) lo que hace mantener su independencia al implantarlo en otras plataformas con diferentes estándares tanto públicos como propietarios.

Al usuario se le proporciona mayor satisfacción cuando el sistema se adapta a él y no es el usuario el que tiene que adaptase al sistema y además permite que cualquier usuario pueda utilizarlo.

6 Proyecto EU4ALL y el Módulo de personalización de contenido

6.1 Descripción del proyecto EU4ALL

En una economía basada en el conocimiento, la educación y el trabajo, dónde estas necesidades forman parte de la vida de una persona, el aprendizaje es necesario para que la población pueda realizar una profesión. Dentro de este aprendizaje, las tecnologías de la información juegan un papel muy importante en la actualidad. Es necesario que esta información pueda llegar a cualquier persona independientemente de las discapacidades que pueda tener.

Si la tecnología utilizada es inadecuada o insuficiente para proporcionar la información a las personas discapacitadas, éstas se verán excluidas de la interconexión entre la educación y trabajo.

El proyecto europeo EU4ALL [5] [15] avanza en el concepto de Asistencia de Aprendizaje Permanente uniendo tres estrategias:

1. Que la tecnología contenga diversidad de caminos para que la gente interactué con ella, con el contenido y con los servicios que proporciona.

2. Que ésta tecnología sea usada para dar servicios de apoyo a gente con discapacidades

3. Proporcionar servicios de apoyo e infraestructura técnica para la enseñanza, que faciliten el camino a personal administrativo y técnico de instituciones educacionales para que puedan ofrecer su enseñanza y sea accesible a personas con discapacidades.

El fin de EU4ALL es mejorar la eficiencia y eficacia de la implementación de estas estrategias, desarrollando una Arquitectura de Servicios Abierta para la Asistencia de Aprendizaje Permanente.

Para lograr un impacto amplio, el objetivo de EU4ALL no es desarrollar un sistema simple, sino un marco de trabajo basado en estándares que facilite la integración de las metodologías con un amplio rango de sistemas *e-learning*.

Por lo tanto EU4ALL:

- Diseña una arquitectura abierta orientada a servicios para la Asistencia de Aprendizaje Permanente.

- Desarrolla la infraestructura de software para los servicios de la Asistencia de Aprendizaje Permanente, estos son, contenido, soporte y servicios de acceso.

- Proporciona especificaciones y/o estándares técnicos para aplicaciones integradas de Asistencia de Aprendizaje Permanente, con estándares de *e-learning* actuales y nuevos.

- Valida los resultados obtenidos.

Orientados a la personalización de contenido, uno de los objetivos de EU4ALL es proporcionar servicios centrados en el usuario individualmente considerando sus necesidades y preferencias, directrices pedagógicas y un comportamiento de adaptación basado en las interacciones del usuario.

El objetivo principal del proyecto EU4ALL es desarrollar un mecanismo flexible, abierto y estándar basado en una arquitectura de servicios de estándares abiertos para el apoyo al aprendizaje permanente en las instituciones de educación superior para las personas con necesidades especiales, con especial atención a las personas con discapacidad y ancianos. Esta combinación de servicios se integrará en un marco de respuesta a usuario, con implementaciones de sistemas específicas de éstos, en el que garantiza que los servicios prestados son abiertos, seguros, basados en estándares, ínter operables y accesibles.

EU4ALL aspira a poner la educación al alcance de todos. Permitirá a personas con discapacidad y personas mayores, sin exclusión de otros grupos minoritarios, obtener un mejor acceso a la educación. Esto aumentará su potencial y autonomía y les permitirá emprender nuevas actividades profesionales, además podrán disfrutar de los servicios básicos como cualquier otro ciudadano.

Por lo tanto, el alcance de EU4ALL es un subconjunto del concepto general de "Vida Asistida", es decir, con Asistencia de Aprendizaje Permanente (ALL).

Los resultados esperados incluyen[6]:

- Un modelo de servicios unificado, compartido y utilizable para apoyar la igualdad de acceso al EHEA.

- Una arquitectura abierta y extensible de servicios para ALL, que está dispuesta a ayudar a estudiantes y a proveedores de servicios.

- Un repositorio para Asistencia al Aprendizaje Permanente, de ámbito europeo que facilita un entendimiento común de las metodologías de aprendizaje, necesidades de acceso, requisitos cognitivos, procedimientos de evaluación y aprendizaje permanente para necesidades especiales.

Gracias a EU4ALL se podrán cubrir discapacidades como:

- Discapacidad visual.

- Deficiencias auditivas.

- Discapacidad física.
- Alteraciones cognitivas.
- Discapacidades de aprendizaje específicas como la dislexia y discalculia.
- Otros problemas cognitivos como el Alzheimer.

EU4ALL se considera un proyecto integrado, ya que va dirigido a las necesidades de la sociedad en Europa y genera un trabajo importante de investigación y desarrollo tecnológico de recursos y competencias.

Además de tener los objetivos científicos y tecnológicos, EU4ALL también centra sus esfuerzos en la construcción de comunidades y la formación impartida a organizaciones de usuarios, la evaluación con los usuarios finales, difusión de los resultados de los proyectos a sus compañeros y público en general, y la transferencia de conocimientos entre instituciones de educación e institutos de investigación.

El Consorcio EU4ALL tiene una experiencia muy variada e importante participación en instituciones de educación superior y grandes empresas. Esto asegura que los objetivos tecnológicos y científicos específicos se pueden lograr y los resultados pueden realmente beneficiar a los usuarios.

UNED y UKOU son considerados como los principales interesados, ya que son dos de las mayores Universidades a Distancia de Europa.

No se pretende crear otro Sistema de Gestión de Aprendizaje (*Learning Management System "LMS"*), sino un marco basado en estándares en los que diversidad de productos pueden basarse. De hecho, incluye servicios que van a ínter operar con actuales y futuros LMS en términos de estándares.

EU4ALL muestra la forma en que los tres conjuntos de directrices de la W3C WAI, que desempeñan un papel importante en el logro de sitios web accesibles, pueden ser utilizados juntos de manera integrada en un contexto de personalización.

Dentro del proyecto EU4ALL se desarrollan asistentes personales para realizar un seguimiento de las necesidades de los usuarios para prestar servicios personalizados, tanto para las necesidades y preferencias de un usuario particular, como la combinación de ellos y su evolución en el tiempo. Se utilizaran técnicas de aprendizaje automático y toda esta información se guarda en el "modelo detallado de usuario", que puede ser definido mediante el estudio de estándares como IMS LIP.

Se aborda temas de investigación en campos como:

- Servicios web semánticos que añaden metadatos al contenido www, significado y relación de datos.
- Arquitecturas abiertas.
- Diseño para todos (D4all).

- Sistemas de aprendizaje accesibles.
- Gestión del conocimiento.
- Aplicaciones web basadas en comunidades.
- Colaboración en educación.

EU4ALL proporcionará nuevos avances tecnológicos con la validación a gran escala que ilustran la forma en que la incorporación de la accesibilidad se puede lograr en las universidades. En particular, en EU4ALL los escenarios se centran en abordar la educación y formación para los estudiantes adultos con necesidades especiales, desde la percepción de movilidad reducida, a discapacidad del lenguaje y cognitiva.

6.2 Módulo de personalización de contenido

El objetivo de la personalización de contenido es permitir que los contenidos se transformen de manera que permita a diferentes usuarios de diferentes niveles de discapacidad acceder a la información en un contexto adecuado.

El módulo de personalización de contenido facilitará un medio para entregar contenido a los usuarios sobre la base de sus preferencias individuales.

La personalización es un proceso tanto impulsado por el usuario como por el sistema que proporciona un resultado adaptado. El diseño y la producción se centran en un producto y nunca se consigue una satisfacción completa en cuanto a lo que el usuario necesita o prefiere.

La personalización de contenido proporciona al usuario un grado de control sobre lo que se le va a entregar y por lo tanto ayuda a alcanzar los objetivos propuestos por el usuario. Por lo tanto las limitaciones de la producción y la entrega, por no ser capaz de conocer las necesidades exactas, se pueden superar si este proceso puede satisfacer una gama más amplia de requisitos en tiempo de ejecución, teniendo en cuenta la información relacionada con el contexto, “usuario, dispositivo y contenido”

Personalización podría implicar la selección de alternativas reales que conllevarían procesos que transforman el contenido particular para obtener las otras alternativas. Esto podría funcionar a través de recursos de aprendizaje individual. Por ejemplo, una imagen en html puede ser entregada junto con un atributo ALT que describe la imagen y por lo tanto se proporciona la presentación a un usuario que puede interpretar visualmente la imagen y a un lector de pantalla que puede emitir una descripción de la imagen fonéticamente.

En un mundo ideal, un recurso de contenido tendrá alternativas para satisfacer todas las opciones que puede necesitar cumplir.

La personalización también implica la eliminación de alternativas para que no estén disponibles en la selección de lo que se ofrece al usuario. Es importante tener un

mecanismo por el cual el usuario pueda saber las alternativas que se han excluido. Esto último se conoce como *scrutability* y se refiere a la capacidad de un usuario a interrogar a su modelo de usuario con el fin de comprender el comportamiento del sistema. O sea, los usuarios deberían ser capaces de controlar lo que el modelo de usuario creé acerca de ellos y la personalización de los procesos asociados.

El módulo de personalización de contenido es uno de los aspectos claves del proyecto EU4ALL, dónde participan tres aspectos importantes:

- El repositorio de metadatos de objetos de aprendizaje.
- El modelo de usuario.
- El modelo de dispositivo.

Dentro de EU4ALL la personalización de contenido intenta contemplar dos enfoques para sistemas adaptados.

- Enfoque tradicional, basado en modelos de conocimiento en un campo determinado, impulsado por inteligencia artificial.
- Enfoque Acceso-para-todos, basado en el modelo de especificaciones del usuario.

El enfoque tradicional se basa en el modelo de usuario, que contiene el grado de conocimiento del usuario, sus preferencias y necesidades. En estos sistemas se definen un conjunto de reglas con el fin de predecir el comportamiento de los usuarios y adaptar el sistema en consecuencia. Sin embargo no pueden prever todas las situaciones posibles que tendrá lugar durante la interacción del usuario con el sistema.

Estas reglas consideran las particularidades del proceso de aprendizaje incluidas en los servicios de enseñanza superior, tales como exámenes de adaptación, matriculación y ayuda psicológica.

Los sistemas con enfoque tradicional se basan en una máquina de proceso de aprendizaje y otras técnicas de Inteligencia Artificial (AI) y su objetivo es ser capaz de adaptar el contenido entregado al usuario y el entorno en el que está trabajando.

El enfoque de acceso-para-todos va dirigido a una parte de metadatos que están asociados y describen propiedades de accesibilidad de objetos de aprendizaje y la adaptación de esos objetos, y a otra parte de requisitos funcionales del usuario, normalmente requisitos de visualización, control del interfaz y limitaciones sobre el contenido.

Este enfoque se basa en seleccionar y transformar los contenidos (incluidas las adaptaciones) para que coincida con los requisitos funcionales definidos por el usuario y el entorno del dispositivo.

El CPSM (*Content Personalisation Software Module*) [3] es un núcleo interno del servicio EU4ALL que se utiliza en muchos otros componentes EU4ALL. Su propósito es

contribuir a garantizar que a un usuario se le presenta un recurso que se adapta a sus necesidades y preferencias.

Cuando el usuario solicita un compuesto de recursos, el CPSM podría dividirlo en sus elementos atómicos y volver con elementos que están más adaptados a las necesidades y preferencias de los usuarios. Por ejemplo, si una persona que está con deficiencias visuales solicita una página HTML con texto y un videoclip, el CPSM podría sustituir el video clip con una descripción que pueda ser leída por un lector de pantalla.

La personalización se puede ver de distinta manera cuando se trata de entornos virtuales de enseñanza y aprendizaje (EVE/A), por ejemplo:

- Personalización del interfaz de usuario.
- Selección de contenidos en base al conocimiento del usuario.
- Adaptación de contenidos en base al conocimiento del usuario.
- Selección de contenidos en base a las preferencias de accesibilidad de los usuarios.
- Adaptación de contenidos en base a las necesidades del usuario, preferencias y dispositivos.

En el Módulo de Software de Personalización de Contenido (CPSM) de EU4ALL se tratan los dos últimos puntos; seleccionar y adaptar los contenidos en base a las necesidades y preferencias de accesibilidad de los usuarios y las características del dispositivo utilizado.

El CPSM es un servicio interno que se comunica con otros servicios EU4ALL y no se pretende que interactué directamente con el usuario. La diferencia para el usuario es que los recursos que se le muestren deben reflejar más sus necesidades y preferencias que si estuviese usando un VLE fuera del marco EU4ALL.

Existen tres entidades necesarias para la realización de la Personalización de Contenido (CP), estas son:

- Modelo de usuario. Que guarda todas las necesidades personales y preferencias del usuario.
- Modelo de dispositivo. Guarda información sobre los dispositivos que se puede utilizar.
- Repositorio de metadatos de los Objetos de Comunicación o aprendizaje.

El CPSM tratará con los metadatos asignados a estas tres entidades para seleccionar y posiblemente adaptar los recursos que respondan a las necesidades y preferencias de los usuarios.

Es importante tener en cuenta cómo debe comportarse el CPSM cuando hay más de una alternativa adecuada de recursos con el mismo grado de idoneidad o como debe comportarse cuando no existe una alternativa adecuada de recursos.

Los pasos necesarios para presentar a un usuario los recursos disponibles más adecuados son:

1. Se envía una petición de usuario al CPSM para un recurso.
2. El CPSM hace un chequeo para ver si hay más alternativas validas de recursos disponibles y si es así selecciona el más adecuado.
3. El CPSM adapta el recurso más adecuado.
4. El CPSM devuelve el recurso al usuario.

7 Subsistema de Gestión de Fachadas

El subsistema de gestión de fachadas consigue abstraer las distintas especificaciones que puedan estar definidas, proporcionando un mecanismo que permite obtener una especificación general, útil para cualquiera de las especificaciones independientes que se quieran utilizar.

El subsistema de gestión de fachadas va a ser aplicado dentro del ámbito del módulo de personalización de contenido. En el siguiente gráfico podemos ver la utilidad que va a tener el subsistema facilitando tanto la representación abstracta para el modelo de usuario, como para el modelo de dispositivo y los objetos de aprendizaje.

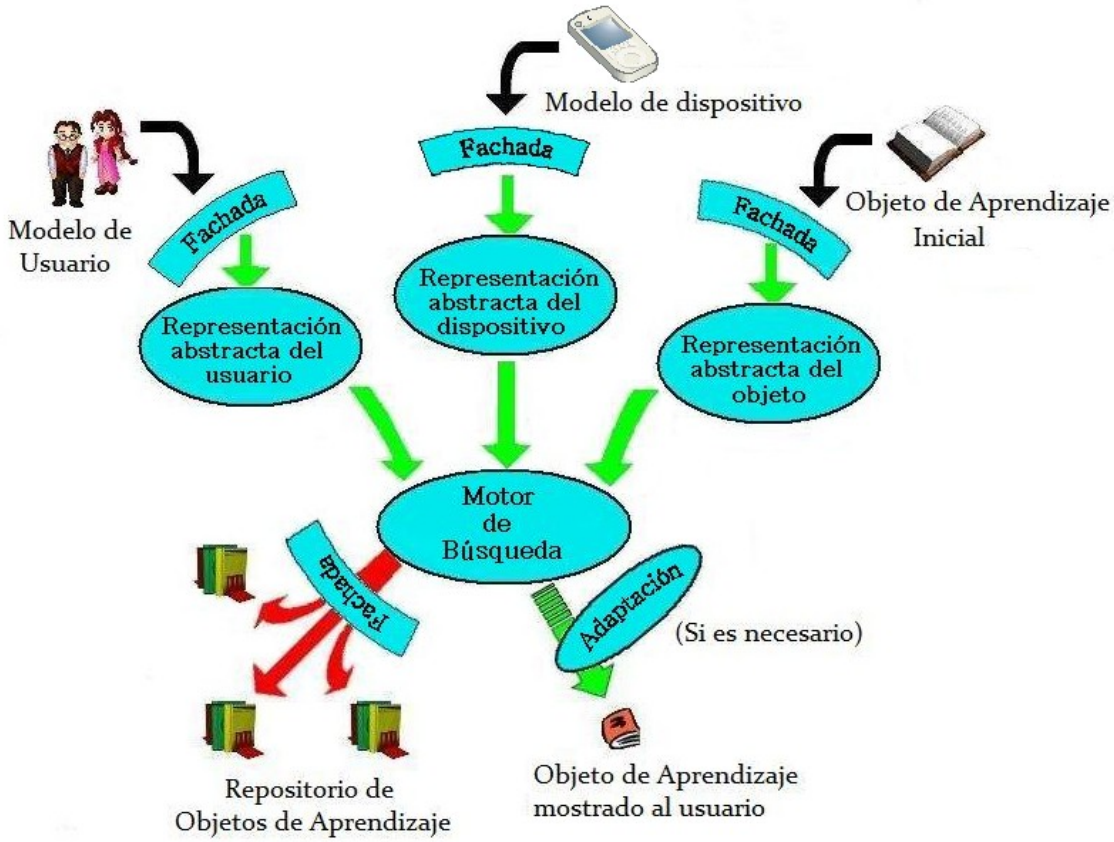

Figura 7 - Servicio de Personalización de contenido

El servicio de personalización de contenido va a recibir el perfil del usuario y un objeto de aprendizaje inicial o recurso. Los pasos a realizar son:

1. Determinar las principales características del objeto de aprendizaje que se quiere mostrar al usuario.
    - Tipo de objeto; imagen, texto, video, etc. Por ejemplo,
        - si es un video: duración, formato, si incluye o no subtítulos,

- si es una imagen: resolución, tamaño, colores, etc.

- Título y Descripción.

2. Obtener del perfil de usuario sus principales características, necesidades, intereses, etc.

3. Obtener las principales características de los dispositivos que el usuario está usando, que vendrán indicadas en los datos del perfil de usuario.

   - *Hardware*: PC, portátil, móvil, PDA…

   - Pantalla táctil, teclado, ratón, lector de pantalla,…

4. En base a esta información obtener el objeto de aprendizaje que satisfaga la información del perfil del usuario y sea lo más equivalente posible al objeto de aprendizaje inicial.

El subsistema de fachadas por lo tanto se va a poder aplicar a la interpretación de perfiles de usuario proporcionando la representación abstracta del usuario, a las características de los objetos de aprendizaje iniciales y de los repositorios de objetos de aprendizaje y al modelo de dispositivo para interpretar las características de los dispositivos utilizados por el usuario.

Adaptando el subsistema de fachadas a la interpretación del modelo de usuario vamos a obtener un perfil de usuario con las características del perfil que únicamente son requeridas para la Personalización de Contenido.

Adaptando el subsistema a la interpretación del modelo de dispositivo obtendremos un interfaz del perfil de dispositivo que nos permite obtener las características que únicamente son necesarias para la Personalización de Contenido.

De igual manera adaptando el subsistema a las características de los objetos de aprendizaje vamos a obtener un interfaz para recuperar la información que interese para la personalización, tanto del objeto inicial como de los objetos de aprendizaje del repositorio, que en este último caso, nos permitiría elegir el recurso más adecuado a partir de ciertos criterios.

7.1 Descripción de la interpretación de perfiles (usuario, recursos, dispositivos)

El subsistema de gestión de fachadas consigue, como objetivo principal dentro de la interpretación de perfiles, abstraer la representación interna que tenga un sistema, más allá de las especificaciones y estándares que existen actualmente para tratar perfiles de usuario.

El modelo de datos del Módulo de Software de personalización de contenido incluye los siguientes submodelos a los que se les va a poder adaptar el subsistema de fachadas [4].

Modelo de Usuario. Este modelo contiene información relacionada con las necesidades de los usuarios y preferencias, edad, idiomas, etc. El Modelo de usuario podría representarse con los estándares:

- *IMS LIP + IMS AccLIP*
- *ISO Individualized Adaptability and Accessibility in E-learning, Education and Training (Type A)*

Modelo de Dispositivo. Almacena la información relacionada con el *hardware* que el usuario está usando y el software instalado en dicho hardware. Algunos de los atributos almacenados en este modelo son:

- *Hardware* (PC, portátil, móvil, PDA ...) y su especificación (CPU, memoria ...)
- Dispositivos (pantalla, teclado, ratón, táctiles, braille ...)
- Conectividad de red, software instalado (la emulación del ratón, lector de pantalla, navegador, java ...)

El modelo de dispositivo podría estar basado en la CC/PP y normas UAProf, con la posibilidad de incluir parte de la IMS a nivel AccLIP como mejor apoyo en el proceso de correspondencia.

MOMR. Almacena la información relacionada con cada recurso en el repositorio de contenido. Podría estar basado en los siguientes estándares:

- *IEEE LOM*
- *Dublín Core [10]*
- *IMS AccMD.*
- *ISO Individualized Adaptability and Accessibility in E-learning, Education and Training (Type B)*

El sistema recibe una representación de modelo de usuario con unas características propias de un estándar concreto. El subsistema de fachadas identifica el servicio que es capaz de interpretar ese modelo de perfil y proporciona un interfaz abstracto que permite al sistema aislarse de ese tratamiento interno para obtener la información. De esta misma manera funcionaría el subsistema de fachadas para el modelo de dispositivo y el modelo de objetos de aprendizaje o recursos.

El grafico 7.1 muestra una visión del proceso de personalización de contenido.

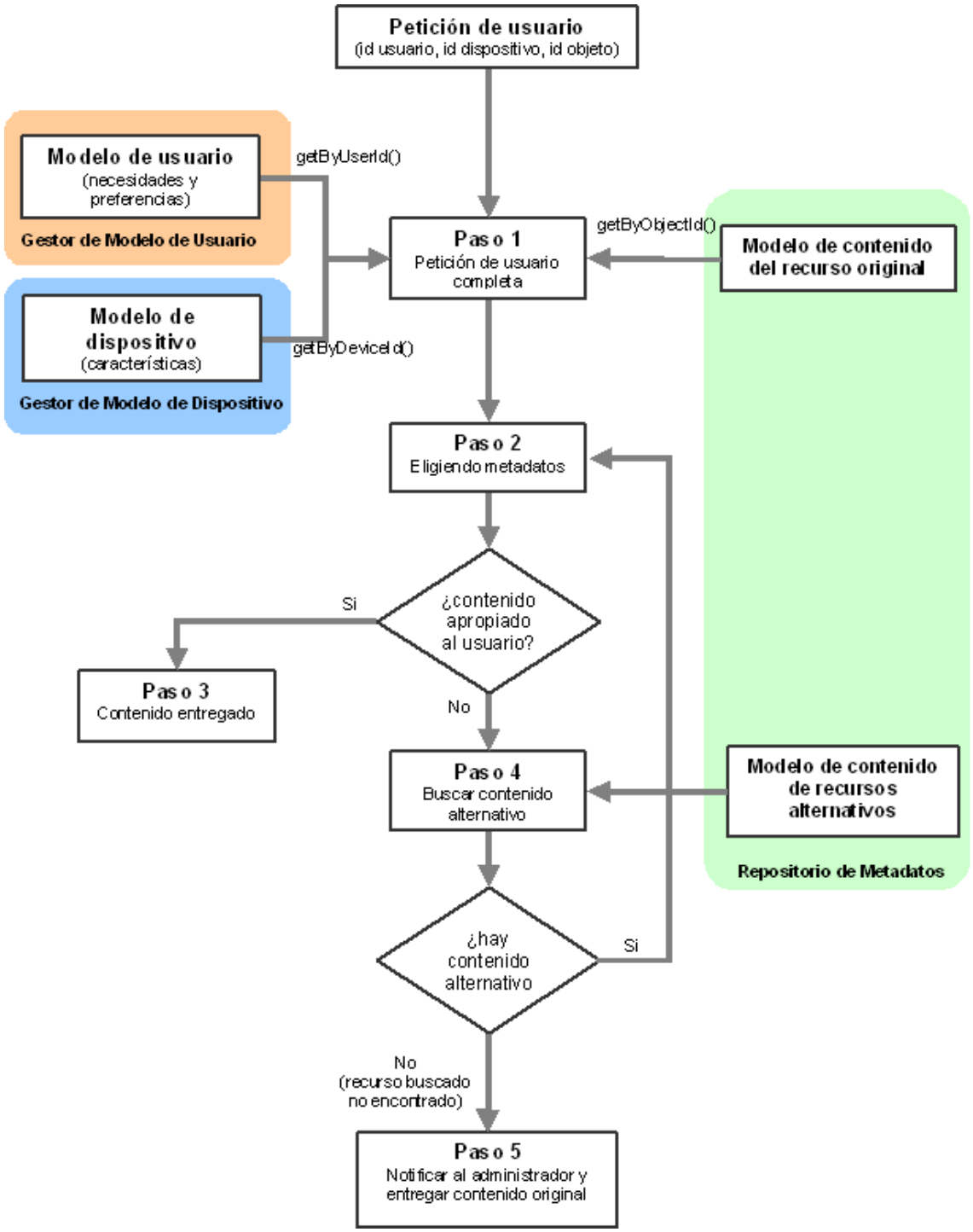

Figura 7.1 - Flujo del Proceso de Personalización de Contenido

El proceso se inicia con la necesidad de adaptar cierto contenido a un usuario. El flujo comenzaría con la llegada de la petición para un usuario que es necesario que incluya,

- la identificación del recurso de contenido original,
- la identificación del usuario,
- la identificación de los tipos de dispositivos que se usan.

El CPSM a continuación deberá:

- Examinar el modelo de usuario para ver las necesidades personales y preferencias, lo que será clave para determinar si un recurso es o no útil para el usuario. Debido a que la representación puede venir en distintos estándares, aquí toma importancia el subsistema de fachadas para la interpretación de perfiles de usuario.

- Examinar el modelo de dispositivos para obtener información sobre los dispositivos que se están usando y poder determinar si un recurso va a poder ser accesible por el usuario. La adaptación del subsistema de fachada a las características de los diferentes dispositivos permitirá obtener la información para evaluar los objetos de aprendizaje.

- Examinar el modelo de contenido para obtener la información sobre el recurso original. Esto es una interfaz con el MOMR que vendrá apoyada en el subsistema de fachadas adaptado a las características de los diferentes objetos de aprendizaje

Se utilizará la información obtenida para determinar si el recurso solicitado se adapta a las necesidades del usuario y los dispositivos usados. Si este no cumpliese cierto grado de validez, se buscarían recursos alternativos en el MOMR volviéndose a realizar el análisis para dar con el más idóneo.

En caso de no obtener una alternativa válida se optaría por el recurso original.

El CPSM podría manipular el recurso para adaptarlo más a las necesidades, pero no es fácil realizar la adaptación de ciertos recursos o incluso hay recursos que no se podrían adaptar.

8 Arquitectura del Subsistema de Fachadas

El objetivo principal del proyecto es realizar un estudio y desarrollo de lo que hemos llamado Subsistema de Gestión de Fachadas dentro del módulo de Personalización de Contenido y más concretamente orientado a la Interpretación de Perfiles de Usuario.

Para mostrar el diseño del proyecto vamos a utilizar notación UML (*Unified Modeling Language*) [12] [13] [14] y la implementación del código resultante se realizará en el lenguaje Java, ya que es uno de los requisitos que nos plantea el proyecto europeo EU4ALL.

8.1 Metodología de desarrollo

Para el proceso de desarrollo se ha tomado de referencia la metodología de desarrollo de Software RUP (*Rational Unified Process*) [25] que junto con el lenguaje de modelado UML para su representación, constituye la metodología estándar más utilizada para el análisis, diseño e implementación de sistemas orientados a objetos.

En el siguiente diagrama se muestra cada una de las fases y elementos que se definen en la metodología de desarrollo utilizada.

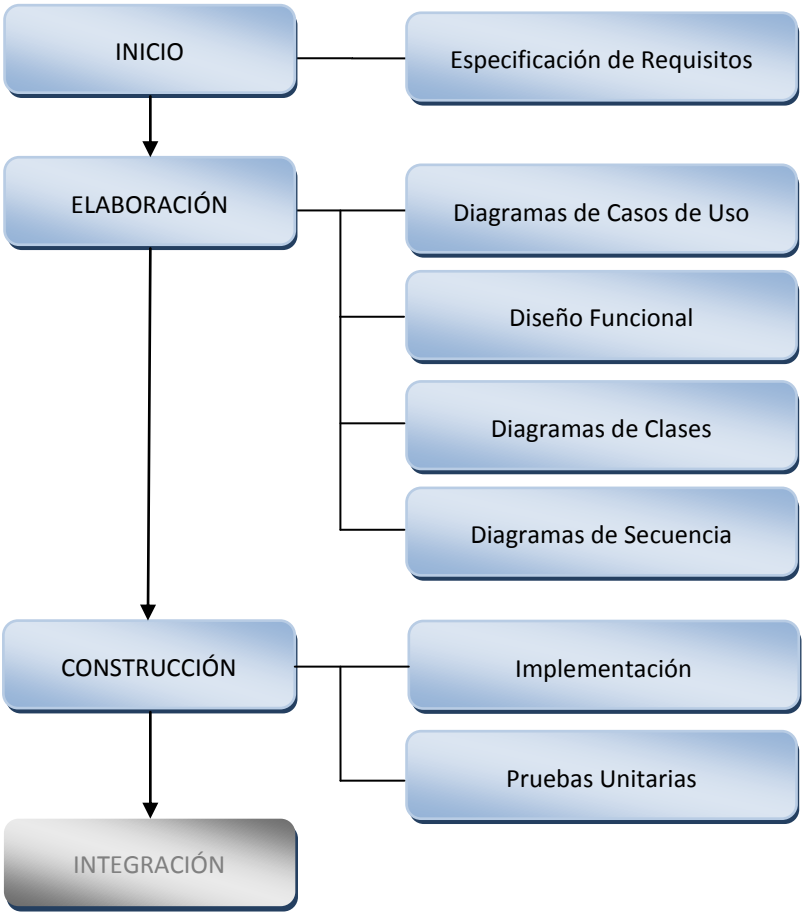

Figura 8.1 – Metodología de desarrollo

En la fase de inicio se definirá el alcance con la toma de requisitos del sistema.

En la fase de elaboración se realiza el análisis y diseño con la especificación de los casos de uso para mostrar el comportamiento y la representación de la estructura e iteración mediante los diagramas de clases y de secuencia.

En la fase de construcción se ejecutará la implementación del software y las pruebas de los casos de uso seleccionados.

En fase de integración el subsistema es entregado al sistema global dónde se integrará en el módulo de personalización de Contenido. No forma parte de éste PFC la ejecución de ésta fase, ya que su competencia fue asumida por el proyecto global.

8.2 Requerimientos del subsistema de fachadas para la interpretación de perfiles

Los requerimientos que se plantean del subsistema de fachadas van orientados hacia la interpretación de los distintos estándares de modelado de datos del perfil de usuario, pero debe tener un diseño lo bastante genérico que permita ser adaptado tanto para otros subsistemas dentro del módulo de personalización de contenido como fuera de él. La necesidad es crear un marco de desarrollo genérico con una adaptación a la interpretación de perfiles de usuario.

Se proporcionará un subsistema de gestión de fachadas con implementaciones concretas para cada uno de los estándares que quieran abordar.

Se quiere dotar al subsistema de robustez, rendimiento y escalabilidad a través del esquema de desarrollo.

- Robustez para soportar fallos parciales.
- Rendimiento para soportar la carga para la que se diseña y dar tiempos de respuesta óptimos.
- Escalabilidad para soportar crecimientos sobre su diseño.

El subsistema va a permitir la interpretación de cualquier estándar de modelado de datos del perfil de usuario.

Se necesita un catálogo de datos dónde se declaren los distintos estándares que el subsistema puede interpretar y la implementación que permitirá la interpretación de cada estándar.

El catálogo de declaración de estándares de usuario será único en todo el sistema por lo que se cargará solo una vez, aunque se podrán realizar recargas para resolver problemas de declaración del mismo.

Sobre el conjunto de información que representa cada estándar de modelado de datos de usuario, el módulo de personalización de contenido tiene definido una especialización de los datos del perfil de usuario que son necesarios para su trabajo.

El subsistema de fachadas debe proporcionar un interfaz genérico que permita obtener esta especialización de datos abstrayéndose de la representación que se este utilizando para ellos. Es decir, que si un usuario se identifica en el sistema utilizando cualquier estándar de modelado de datos reconocido por el subsistema, el módulo de personalización de contenidos a través del subsistema de gestión de fachadas obtendrá un objeto genérico que le permita acceder a esos datos.

También deberá poder acceder, si lo desea, a la representación específica de todos los datos de cada estándar de perfil de usuario.

Si surgiese la necesidad de añadir a la especialización la obtención de nuevos datos de usuario, los cambios a realizar deberían afectar lo menos posible al subsistema.

El subsistema debe poder crecer fácilmente, permitiendo incorporar nuevas implementación de estándares de modelado de usuario sin que esto le suponga tener que adaptarse.

El subsistema proporcionará interfases para facilitar su uso por otros componentes del sistema o del marco de desarrollo.

- Interfaz de inicialización del subsistema. Permite lanzar la inicialización del subsistema. Realizará el control de que el subsistema sea inicializado una única vez.

- Interfaz de cargar del catálogo. Permite realizar la carga/recarga del catálogo de fachadas o servicios.

- Interfaz para obtener la fachada correcta. Las solicitudes deberán incluir dos elementos clave, el documento con los datos de usuario y la clave del estándar de modelado de perfil de usuario.

- Interfaz para obtener la representación del documento del perfil de usuario, que permitirá recuperar cualquier dato del mismo, no únicamente los requeridos por el módulo de Personalización de Contenido. Las solicitudes deberán incluir dos elementos clave, el documento con los datos de usuario y la clave del estándar de modelado de perfil de usuario que se utiliza.

8.3 Casos de uso

8.3.1 Diagrama de Casos de Usos

En este diagrama vamos a ver los distintos casos de uso del subsistema de fachadas. Estos casos de uso representan el comportamiento del subsistema para suministrar al sistema distintas tareas y las relaciones existentes entre cada una de las tareas.

Podemos extraer del subsistema tres casos de uso:

1. Inicializar subsistema. Cargar el subsistema de fachadas con todos los datos necesarios para estar preparado ante una petición.

2. Recargar catálogo. Cargar el catálogo con los datos iniciales o realizar una recarga de los mismos.

3. Obtener servicio. Proporcionar el servicio que se ajusta a la petición realizada.

En la figura 8.3.1 se muestra el diagrama de casos de uso del subsistema de fachadas.

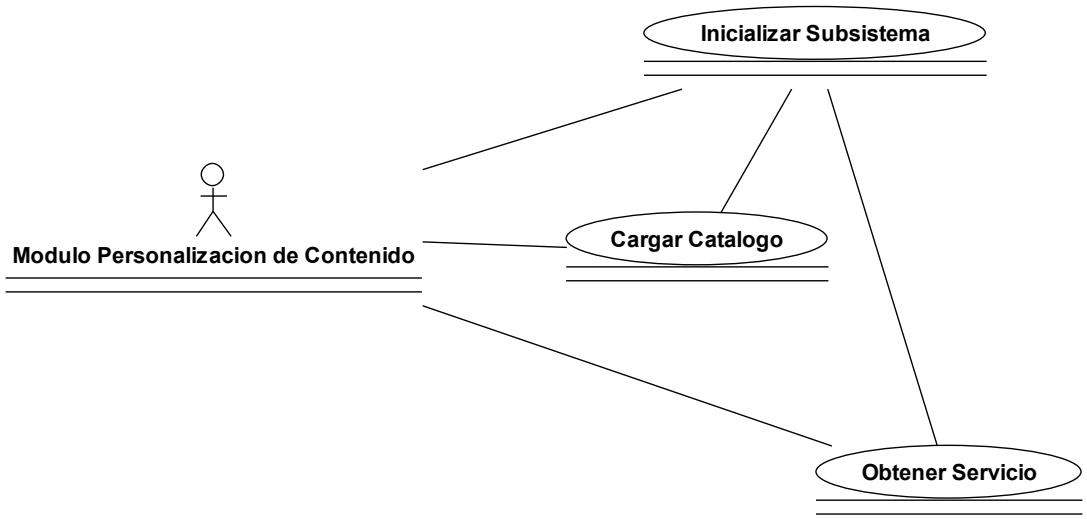

Figura 8.3.1 – Diagrama de Casos de Uso del Subsistema de Fachadas

Como se puede ver en el diagrama, existe un único actor externo que va a interactuar con los casos de uso el subsistema de fachadas, el Módulo de Personalización de contenido, ya que el subsistema es una parte de este módulo y su creación surge como una necesidad de proporcionarle una funcionalidad específica. Aún así este modelo sería válido para aplicarse en cualquier otro módulo del sistema.

La inicialización del subsistema supondrá la creación de todos los elementos y la carga de datos que van a permitir trabajar al subsistema de una forma óptima y eficaz.

La carga del catálogo representa la inicialización de los datos del catálogo de servicios, bien sea en una primera instancia o porque se solicite una recarga posterior ante alguna variación de la declaración de sus datos.

La obtención de servicio permitirá devolver al actor el servicio adecuado para trabajar con la información del usuario que interactúa en el sistema.

Podemos apreciar que existen relaciones entre los distintos casos de uso identificados.

Existe una relación entre el caso de uso de Inicialización del Subsistema y Carga del Catálogo, ya que la carga del catálogo es una parte importante de la inicialización del

subsistema. A su vez para poder realizar la recarga del catálogo es preciso que el subsistema ya se encuentre inicializado, de no ser así esta petición provocaría la inicialización del subsistema.

También existe una asociación entre el caso de uso de obtención de servicio e inicialización del subsistema, ya que es necesario que el subsistema se encuentre inicializado para gestionar esta petición. En el caso de no encontrarse inicializado, la petición provocaría la inicialización del subsistema antes de obtener el servicio.

8.3.2 Fichas de Casos de Usos

En las siguientes plantillas se muestra el curso básico de los casos de uso que describen la funcionalidad que debe implementar el subsistema y que posteriormente serán descritos con más detalle.

La inicialización del subsistema es imprescindible para su funcionamiento y le va a proporcionara una respuesta óptima de procesamiento.

Nombre de Caso de Uso	Inicializar Subsistema
Objetivos	Inicialización del gestor de fachadas y carga y validación del catálogo
Actores	Módulo de Personalización de Contenido
Precondiciones	Subsistema no inicializado
Postcondiciones	Instancia del gestor de fachadas inicializada Catálogo cargado
Escenario básico	- Solicitud de inicialización - Inicialización del gestor de Fachadas - Carga y validación del catálogo

La carga del catalogo además de ser necesaria para la inicialización va a permitir un crecimiento dinámico del subsistema permitiendo recargas del mismo.

Nombre de Caso de Uso	Carga del catálogo
Objetivos	Cargar los datos del documento del catálogo validando su definición

Actores	Módulo de Personalización de Contenido
Precondiciones	
Postcondiciones	Catálogo cargado
Escenario básico	- Solicitud de carga - Carga y validación del catálogo

La petición de servicio va a permitir obtener la fachada del subsistema ya inicializado.

Nombre de Caso de Uso	**Petición de servicio**
Objetivos	Obtener el servicio que interpreta los datos del usuario
Actores	Módulo de Personalización de Contenido
Precondiciones	Documento de datos del usuario Clave del estándar identificada Subsistema inicializado
Postcondiciones	Se obtiene la fachada para los datos del usuario
Escenario básico	- Petición de servicio - El gestor de fachadas devuelve el interfaz fachada con el servicio interprete

8.3.3 Caso de Uso de Inicialización del Subsistema de Fachadas

El subsistema de fachadas debe proporcionar funcionalidad para poder inicializarse en el momento que sea necesario utilizarlo.

La petición va a ser realizada por el módulo de personalización bien sea directamente o a través de otra petición al subsistema, ya que para que el subsistema pueda resolver cualquier petición es necesario que haya sido inicializado previamente.

La inicialización, supone la creación de todos los elementos del subsistema, y la validación y carga de datos.

Los elementos más importantes del subsistema serían:

- Gestor de fachadas, que es el encargado de recibir y resolver las peticiones.

- Catálogo, que guarda las clases de los servicios y representaciones de los datos.

El subsistema únicamente será inicializado una vez para proporcionar un rendimiento más óptimo en su funcionamiento.

Dentro del este caso de uso nos encontramos con distintos escenarios:

1. El Módulo de Personalización de Contenido, solicita la inicialización del subsistema por primera vez. Esto va a provocar la creación de todos los elementos y carga de datos necesarios para un óptimo funcionamiento del subsistema. Una de las partes importantes de esta inicialización va a ser la carga del catálogo.

2. Es otro caso de uso el que solicita la inicialización del subsistema por primera vez, para resolver la petición. Al igual que en el punto anterior, va a provocar la creación de todos los elementos y carga de datos necesarios para un óptimo funcionamiento del subsistema. Una vez inicializado se dispondrá del gestor de fachadas para resolver la petición.

3. La solicitud de inicialización no supone la inicialización del subsistema, tanto si la petición procede del Módulo de Personalización de Contenido, como si de otro caso de uso. En este casó el actor obtendría el gestor de fachadas ya inicializado anteriormente para poder trabajar con el subsistema y resolver sus peticiones.

4. En el caso de que se produjese algún error de declaración en el catálogo que no permitiese la carga del mismo, no se completaría la inicialización del subsistema lo que generaría un error y en una petición posterior, una vez resuelto el problema, volvería a inicializarse por completo.

Como se puede apreciar, no es necesaria la solicitud explicita por parte del Módulo de personalización de contenido para que el subsistema sea inicializado, sino que cualquier petición al subsistema puede ser la que provoque esa inicialización, previa a la resolución de la petición.

De esta manera vamos a aislar el funcionamiento interno del subsistema del exterior.

8.3.4 Caso de Uso de Carga del Catálogo del Subsistema de Fachadas

El catálogo es una parte muy importante del subsistema que debe tener los datos cargados para poder atender a las peticiones que el subsistema recibe de sus actores.

A priori el catálogo va a cargarse una única vez cuando se realice la inicialización del subsistema de fachadas y no cada vez que reciba una petición. De esta manera se consigue optimizar el tratamiento.

Vamos a considerar que el catálogo puede sufrir modificaciones o queremos poder actuar ante algún error de declaración de datos. Para suplir estos casos, se va a proporcionar flexibilidad al subsistema permitiendo realizar recargas del catálogo cuando sea necesario.

Podemos diferenciar varios escenarios:

1. El catálogo se va a cargar inicialmente en el momento que el subsistema es inicializado. Esta carga supone guardar las declaraciones del catálogo, validar los datos y crear las implementaciones de los servicios e interfaces de contenido definidas.

2. El catálogo va a ser recargado de nuevo para actualizar sus datos. Los datos anteriormente cargados no serán validos y se volverán a cargar las declaraciones del catálogo, validar los datos y crear las implementaciones de los servicios y representaciones de contenido definidos.

3. Se genera un error al realizar la carga inicial del catálogo. Al solicitar la carga del catálogo se realiza la validación de los datos que se encuentran definidos, comprobando que existe la implementación de las clases que se indican. Si no se cumple alguna de las validaciones, el catálogo no sería cargado generando un error de carga.

4. Se genera un error al realizar una recarga del catálogo. Al solicitar una recarga del catálogo se realiza la validación de los datos que se encuentran definidos, comprobando que existen la implementación de las clases que se indican. Si no se cumple alguna de las validaciones, se va a generar un error de carga pero el catálogo conservará los datos con tenia anteriormente, de esta manera una modificación errónea en la declaración de datos no afectará al funcionamiento del subsistema hasta que éste error sea corregido.

En definitiva el catálogo será cargado a la hora de inicializarse el subsistema y podrá ser recargado cuando exista alguna necesidad, sin que un error en la actualización del catálogo provoque un fallo en el funcionamiento del subsistema.

8.3.5 Caso de Uso de Petición de un Servicio al Subsistema de Fachadas

El módulo de personalización de contenido se va a encargar de realizar peticiones al subsistema de fachadas para obtener los servicios que le permitan interpretar el perfil del usuario con el que interactúa.

Este caso de uso representa como el subsistema de fachadas va a obtener el servicio apropiado para interpretar el perfil del usuario identificado.

Para poder resolver la petición es necesario que el subsistema de fachadas se encuentre inicializado.

Nos encontramos con los siguientes escenarios:

1. En el momento de recibir la petición, el subsistema validará si no ha sido inicializado, en cuyo caso se lanzará la inicialización y posteriormente se tratará la petición. En caso de que la inicialización generase un error, la petición no podría ser resuelta.

2. El subsistema ya se encuentra inicializado, por lo que se realiza la petición del servicio al gestor de fachadas obteniendo el servicio capaz de interpretar el perfil de usuario.

3. El servicio solicitado no se encuentra definido en el catálogo, lo que va a generar un error que le será indicado al Módulo de personalización de contenido.

8.4 Diseño Funcional

En la figura 8.4 podemos ver un esquema en el que se representa el diseño del subsistema de fachadas.

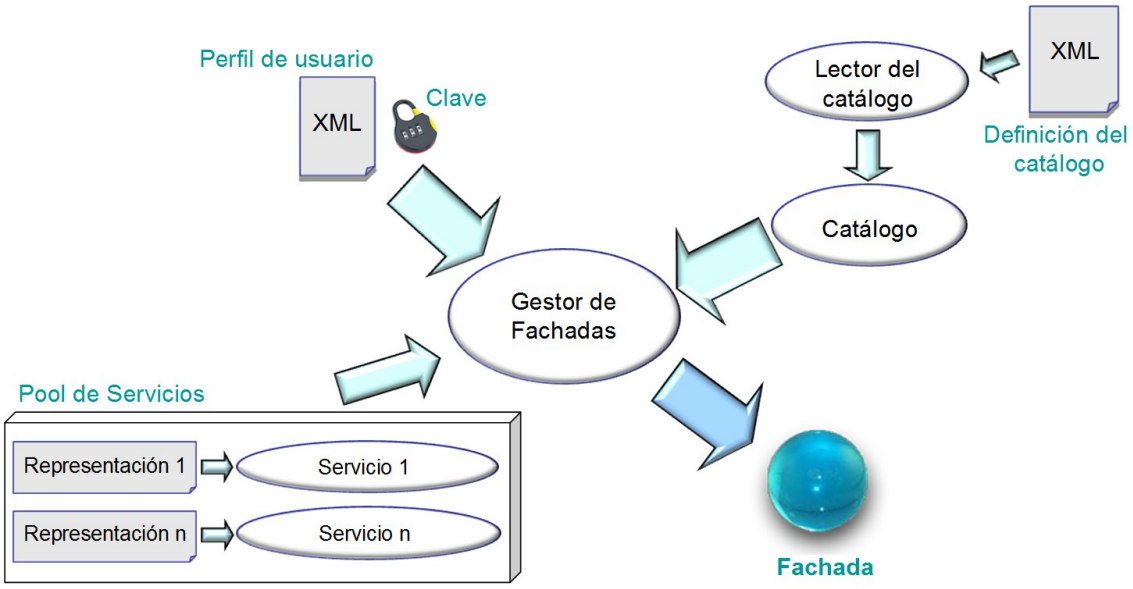

Figura 8.4 - Esquema de Diseño del Subsistema de Fachadas

La representación de los documentos con la definición del catálogo y de los datos de los perfiles de usuario será en XML.

El lector de catálogo se encargará de leer la declaración de las representaciones y servicios con los que el catálogo se inicializará.

El gestor de fachadas recibirá el documento XML con una representación de datos determinada, lo que para interpretación de perfiles seria el estándar de modelado de

usuario y también recibirá la clave que identifica dicha representación del documento recibido.

El catálogo le indicará al gestor de fachadas la clase adecuada para leer la representación de los datos y el servicio adecuado para interpretarlos.

El gestor de fachadas obtendrá del pool de servicios la instancia de la representación y el servicio, capaces de interpretar el documento con los datos de usuario en el estándar indicado.

El resultado es un interfaz genérico con la implementación adecuada para poder interpretar los datos.

8.5 Diseño Técnico

En el diseño técnico vamos a ver en detalle el marco de desarrollo que constituye el subsistema de fachadas.

Vamos a mostrar las normas que se deben cumplir para un funcionamiento correcto y un método sencillo de uso.

Se puede apreciar de qué forma el subsistema tiene capacidad para cambiar, crecer y ser utilizado en diferentes partes del sistema global.

Para realizar el diseño técnico del Subsistema de gestión de fachadas, vamos utilizar como notación el lenguaje UML y mediante un conjunto de diagramas representaremos y explicaremos el funcionamiento del subsistema.

Los diagramas a representar serán:

- Diagramas de Clases de Diseño, que describe la estructura del subsistema, mostrando las clases de las que se compone y las relaciones estáticas que existen entre ellas.

- Diagramas de Secuencia, que representa la interacción entre los objetos, pero con aspecto temporal.

8.5.6 Patrón de diseño Facade

Pongámonos en el ejemplo de un sistema en el que algunos usuarios con problemas visuales que les impide leer bien las fuentes pequeñas tendrían que aumentar el tamaño de las fuentes. Esto fuerza al usuario a que recorra todos los pasos de configuración, con el tamaño de fuente actual, hasta llegar a la opción que necesita, lo que no que sería muy agradable para él. Tener un asiste que ayude a realizar la configuración a personas con problemas visuales, sería mucho mejor.

Esta ayuda no limitaría el resto de opciones, por lo que se proporcionaría una vista especializada del sistema y al mismo tiempo, se mantienen el resto de características. Este tipo de patrón *Facade* [1] es lo que se llamaría un *front-end*, o asistente, para el sistema.

En definitiva, el propósito de este patrón es proporcionar una interfaz simplificada para un grupo de subsistemas o sistema complejo.

La solución de diseño que se plantea para desarrollar el subsistema de fachadas viene basada en este patrón de diseño.

El subsistema de fachadas va a proporcionar un interfaz sencillo o Fachada con una especialización de funcionalidad que facilitará el trabajo.

La implementación del patrón Facade se compone de:

- La Fachada. Que es la clase que utilizan los clientes. Conoce los subsistemas y sus responsabilidades. Las peticiones serían delegadas al subsistema que corresponda

- El Subsistema. Es un conjunto de clases que pueden ser utilizadas directamente por los clientes o realizar el trabajo que les solicita el objeto Fachada.

Inicialmente para nuestra implementación la adaptación de este patrón se podría ver de tal manera que la clase Fachada seria lo que llamamos el Gestor de Fachadas que estará representado en la clase *FacadeManager*, y las clases del subsistema serían cada una de las representaciones de datos definidas.

El gestor de fachadas debería definir todos los métodos que permiten obtener los datos necesarios y en cada petición resolver la representación a la que dirigirse y la forma de obtener los datos.

En el diseño del subsistema existirá un servicio por cada clase de representación de datos que tendrá definida la especialización de los mismos, o sea, la obtención de los datos más relevantes o lo que llamaremos, interpretación de los datos. Por lo tanto cada servicio sería la fachada de la representación de los datos.

Para cumplir con los requisitos del subsistema en cuanto a rendimiento, el gestor de fachadas, como su propio nombre indica, no se encargará de resolver cada una de las peticiones de obtención de datos, sino que va a proporcionar el servicio o fachada adecuada para que el sistema trabaje directamente con ella.

La especialización de todos los servicios va a ser la misma, por lo que el subsistema va a proporcionar un mecanismo mediante el cual el actor trabajará con una única fachada.

8.5.7 Diagramas de clases del subsistema de fachadas

En el siguiente apartado hablaremos de todas las clases que forman el subsistema de fachadas y mostraremos mediante unos diagramas las clases más importantes del subsistema que nos permitirán ver y entender como está estructurado.

Dentro del subsistema nos podemos encontrar con las siguientes clases:

- Un conjunto de interfaces.
  - Interfaz del catálogo.
  - Interfaz para leer el documento del catálogo.
  - Interfaz de los servicios que interpretan los estándares de los perfiles de usuario.
  - Interfaz de los lectores de los documentos de datos de los perfiles de usuario.
- Gestor de Fachadas, que es la conexión del subsistema con el exterior ya que se encarga de recibir y gestionar las peticiones.
- Clases que implementan las interfases.
  - Implementación del catálogo de servicios para una representación XML.
  - Implementación del lector del documento del catálogo en una representación XML.
  - Implementación base de los servicios que interpretan los estándares de perfiles de usuario para una representación XML.
  - Implementación base de los lectores de los documentos de datos de los perfiles de usuario.
- Clases que representan el catálogo.
  - Estructura que representa los datos de un elemento del catálogo.
  - Representación de la clave de los elementos del catálogo. Se usa para poder identificar un elemento del catálogo.
- Clases de ayuda para facilitar el tratamiento del subsistema.
  - Control de excepciones del subsistema.
  - Manejo del XML que representa los datos.

Mediante este conjunto de clases vamos a proporcionar un subsistema fácil de utilizar, flexible a cambios y con un alto grado de reusabilidad.

Las implementaciones de las clases van dirigidas al tratamiento de documentos con formato XML ya que tanto el documento con la definición del catálogo como los documentos con los datos de los perfiles de usuario, van a venir representados en formato XML.

Las clases de ayuda son clases de utilidades que existen en el subsistema y que son usadas por las clases principales. Estas clases encapsulan funcionalidad que proporciona el lenguaje java y se han desarrollado como una utilidad para facilitar el desarrollo del subsistema, proporcionando un interfaz más sencillo para una funcionalidad más compleja. La representación de estas clases no la mostraremos en los diagramas.

8.5.8 Diagrama de clases del marco de desarrollo del Subsistema de Fachadas

En este diagrama se representa el núcleo central de desarrollo del subsistema de fachadas.

Los dos componentes más importantes del núcleo son el gestor de fachadas y el catálogo, ya que el primero va a ser el encargado de gestionar y resolver las peticiones que recibe el subsistema y el segundo el que guarde la información necesaria para que las peticiones se resuelvan correctamente.

Dentro del marco de desarrollo, existen otros componentes igualmente importantes que van a facilitar al gestor de fachadas y al catálogo la resolución de las peticiones, estos son:

- El lector del documento que contiene la definición del catálogo.
- La estructura que representa un elemento del catálogo.
- La representación de la clave con la que se van a identificar las declaraciones del catálogo y las peticiones al gestor de fachadas.

Para completar el marco de desarrollo es necesario incluir la estructura que representa el documento con los datos del perfil del usuario y el servicio que va a interpretar ese perfil de usuario.

En definitiva el marco de desarrollo del subsistema está constituido por los siguientes componentes.

- Gestor de fachadas. El subsistema proporciona un Gestor de Fachadas que va a ser único y representa el nexo de unión del subsistema con el exterior, ya que todas las peticiones que reciba el subsistema pasarán por él y se encargará de gestionarlas. Este Gestor de Fachadas está representado con la clase FacadeManager, que cumple el patrón singleton, ya que no podrá existir más de una instancia en el sistema.

- Catálogo. Contiene toda la información que le va a permitir al gestor de fachadas resolver las peticiones que recibe. En el marco de desarrollo, para representar el catálogo, se proporciona una clase *IFacadeCatalog* que es el interfaz que debe cumplir cualquier implementación que se haga del catálogo.

- Lector del documento del catálogo. Es el encargado de interpretar el documento que contiene las declaraciones del catálogo. En el marco de desarrollo lo representamos con el interfaz *IFacadeIdentificationRead* que describe la funcionalidad que debe cumplir cualquier implementación para leer el documento del catálogo. El lector permitirá aislar al resto del subsistema de la manera en que está representado el documento y permitirá cargarlo en una estructura que facilite su manejo.

- Elemento del catálogo. Es una estructura que va a guardar los datos de un elemento del documento del catálogo. Está representada por la clase *FacadeIdentificationStruct*. A partir de esta estructura se podrán almacenar los elemento del catálogo e incluso identificarlos tanto dentro como fuera del subsistema ya que proporciona funcionalidad para obtener la clave.

- Representación de la clave del catálogo. El marco de desarrollo va a proporcionar una clase que guarde el conocimiento de la formación de la clave de un elemento del catálogo. Esta clase es la *FacadeKey*, y está totalmente relacionada con la *FacadeIdentificationStruct*, ya que va a construir la clave a partir de todo o parte de sus datos. Permite trabajar al resto del subsistema con la clave del catálogo independientemente de cómo esté construida.

- Representación del documento del perfil de usuario. Es un interfaz genérico llamado *IFacadeRepresentation*, que se va a utilizar para proporcionar la clase que representa el documento con los datos del perfil del usuario para un estándar. Cada estándar de documento de perfil de usuario tendrá su propio interfaz que debe heredar de éste para que pueda ser tratado por el subsistema.

- Servicio interprete del perfil de usuario. Es el interfaz que debe cumplir cualquier servicio declarado en el catálogo que es capaz de interpretar el perfil de usuario para un estándar dado. Se representa con la clase *IFacadeService*.

En la figura 8.5.8 se muestra el diagrama de clases del núcleo del subsistema.

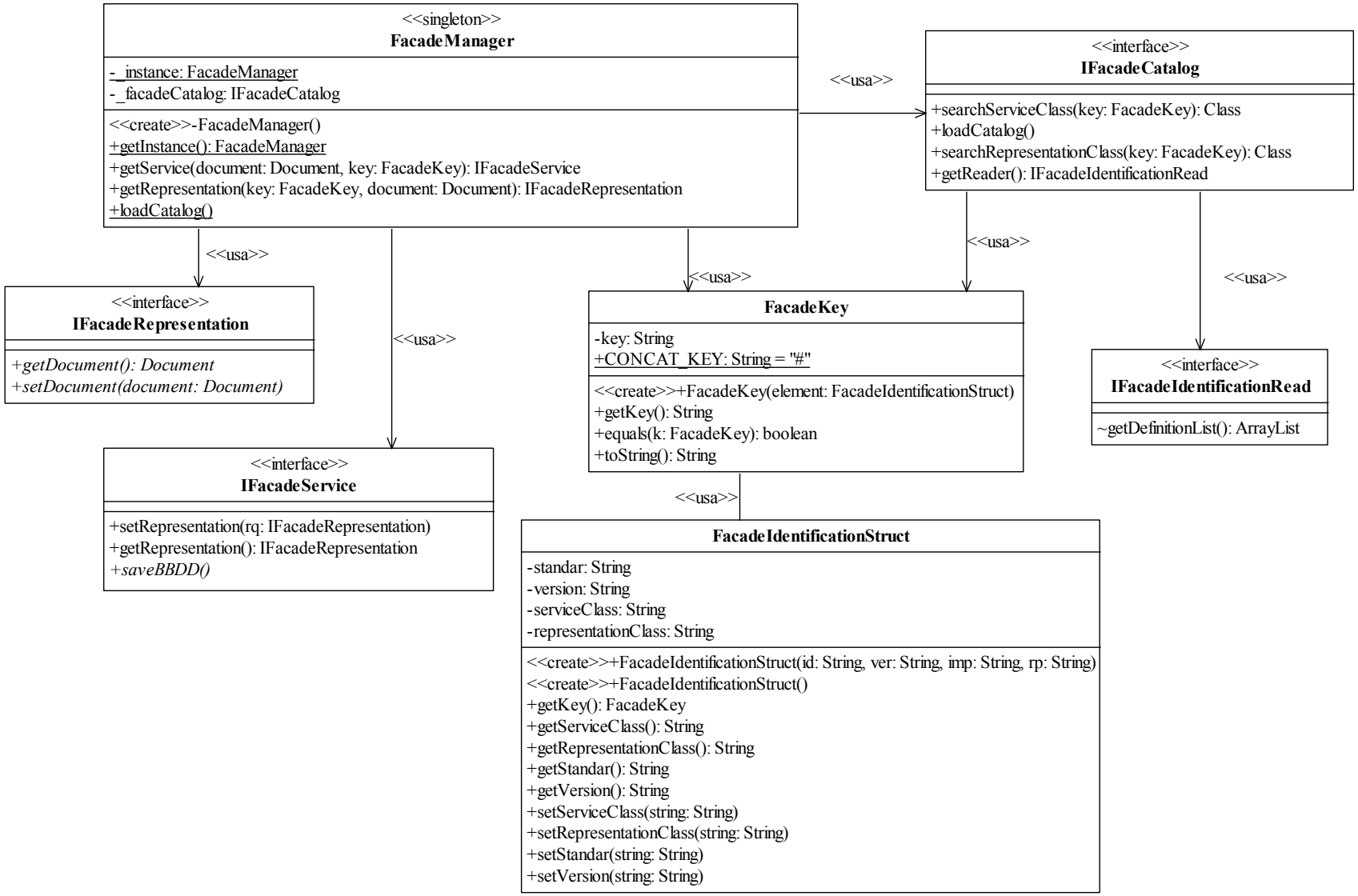

Figura 8.5.8 – Diagrama de Clases del Marco de Desarrollo del Subsistema de Fachadas

La relación existente entre el gestor de fachadas y el catálogo es unidireccional, dónde el gestor realiza solicitudes al catálogo.

El gestor es el único que tiene relación con las clases *IFacadeService* e *IFacadeRepresentation*, ya que el tratamiento de las clases por parte del catálogo se realiza de forma genérica.

La relación existente entre el catálogo, *IFacadeCatalog* y el lector *IFacadeIdentificationRead*, supone que al realizar una implementación del catálogo, debe realizase también la implementación del lector del documento del catálogo que va a usar el catálogo implementado.

La clase clave *FacadeKey* está fuertemente relacionada con el elemento del catálogo *FacadeIdentificationStruct*, ya que necesita conocer de qué está compuesto para saber con que campos va a formar la clave.

Al crear una clase clave *FacadeKey* independiente, que es la única del subsistema que tiene el conocimiento de cómo está formada la clave del catálogo, nos permite independizar del gestor de fachadas y del catálogo la composición de la clave, de manera que si la clave se decidiese representar con otros campos del elemento del catálogo *FacadeIdentificationStruct*, la clase *FacadeKey* sería la única del subsistema que debería adaptase.

Pasamos a describir con más detalle cada una de las clases que intervienen en este diagrama.

FacadeManager

Es el gestor de fachadas que actúa como clase principal del subsistema y proporciona la fachada adecuada para tratar con los estándares de perfil de usuario. Se implementa como una clase singleton por lo que únicamente va a existir una en todo el sistema.

Es el propio gestor el que guarda el catálogo con la relación entre estándares y clases que los resuelven.

Atributos

- FacadeManager *_instance*: único objeto que va a ser creado de ésta clase.
- IFacadeCatalog *_facadeCatalog*: referencia al catálogo del subsistema.

Métodos

- FacadeManager: Constructor privado para asegurar el patrón singleton y que así exista únicamente una instancia del gestor de fachadas. En el momento de generarse el gestor de fachadas se va a lanzar la creación del catálogo.
- getInstance: Crea o devuelve la instancia ya creada del gestor de fachadas.

- Retorno
  - FacadeManager: la propia instancia.

- getService: Proporciona el servicio capaz de interpretar el estándar con el que está representado el perfil de usuario. Debe recibir la clave identificativa del estándar usado y el documento con los datos del perfil de usuario representado en ese estándar.
  - Parámetros
    - Document: Documento XML con el perfil del usuario en un estándar dado.
    - FacadeKey: Clave del catálogo para el estándar utilizado para representar el perfil de usuario.
  - Retorno
    - IFacadeService: Devuelve en esta variable interfaz genérica la instancia de la clase que implementa el servicio que corresponde al estándar identificado.
- getRepresentation: Devuelve la clase que representa el documento del estándar utilizado para el perfil de usuario. Debe recibir la clave identificativa del estándar usado y el documento con los datos del perfil de usuario en el formato de ese estándar.
  - Parámetros
    - FacadeKey: Clave del catálogo para el estándar utilizado para representar el perfil de usuario.
    - Document: Documento XML con el perfil del usuario en un estándar dado.
  - Retorno
    - IFacadeRepresentation: Devuelve en esta interfaz genérica, la instancia de la clase que implementa la representación que corresponde al estándar identificado.
- loadCatalog: Lanza la carga o recarga del catálogo dependiendo de si el catálogo ya se encontraba inicializado. Devolverá error en caso de que no se haya podido cargar, pero si el catálogo ya se encontraba cargado anteriormente, esos datos se conservarán. No tiene parámetros ni retorno.

IFacadeCatalog

Interfaz que debe cumplir cualquier implementación del catálogo del subsistema de fachadas.

Atributos

No tiene atributos

Métodos

- searchServiceClass: Devuelve la clase servicio que se corresponde a la clave identificada. En el caso de no encontrarlo, devolverá una excepción que lo indique.
  - Parámetros
    - FacadeKey: Clave del catálogo del estándar utilizado para representar el perfil de usuario.
  - Retorno
    - Class: Clase del servicio que se corresponde a la clave dada.
- searchRepresentationClass: Devuelve la clase que representa los datos el estándar identificado en la clave. En el caso de no encontrarla, devolverá una excepción que lo indique.
  - Parámetros
    - FacadeKey: Clave del catálogo del estándar utilizado para representar el perfil de usuario.
  - Retorno
    - Class: Clase para leer los datos de perfil de usuario del estándar indicado.
- getReader: Obtiene el lector del documento que contiene los datos del catálogo.
  - Retorno
    - IFacadeIdentificationRead: Devuelve en esta variable interfaz el objeto que implementa la lectura del documento con los datos del catálogo.
- loadCatalog: Fuerza la carga del catálogo. El catálogo se cargará de nuevo comprobando la existencia de las clases indicadas en él. Se devolvería error en caso de producirse, pero los datos del catálogo se quedarían como estaban anteriormente.

FacadeIdentificationStruct

Estructura de datos que guarda la información de un elemento del catálogo. Sus atributos constituyen una entrada del catálogo. Proporciona funcionalidad para poder obtener la representación de la clave.

Atributos

- String *standar*: nombre del estándar de perfil de usuario.

- String *version*: versión del estándar de perfil de usuario.
- String *serviceClass*: nombre de la clase servicio que interpreta el estándar y para esa versión.
- String *representationClass*: nombre de la clase que representa los datos para ese estándar y versión.

Métodos

- FacadeIdentificationStruct: Constructor vacio que permite crear un elemento del catálogo sin inicializar sus datos.
- FacadeIdentificationStruct: Constructor de un elemento del catálogo. Se le debe indicar como parámetros todos los atributos que van a formar un elemento del catálogo.
  - Parámetros
    - String *standar*
    - String *version*
    - String *serviceClass*
    - String *representationClass*
- getKey: Permite obtener la clave de la propia instancia. Se devuelve un objeto FacadeKey que representa la clave.
  - Retorno
    - FacadeKey. Objeto que representa la clave.
- Métodos get y set para cada uno de los atributos de la clase.

FacadeKey

Esta clase representa la clave de un elemento del catálogo. Es la única que conoce como se forma la clave a partir de un elemento del catálogo. No realiza validación de datos nulos o vacíos, por lo que no es responsabilidad suya el proporcionar una cadena representativa de la clave vacía. Proporciona funcionalidad para facilitar el tratamiento con las claves del catálogo y flexibilidad ante cambios.

Atributos

- String *key*: cadena representativa de la clave
- String *CONCAT_KEY*: constante con valor "#" que se utiliza como valor de concatenación para formar la clave si está compuesta por más de un valor.

Métodos

- FacadeKey: Constructor que recibe un elemento del catálogo con el que va a generar la clave. No es necesario que el elemento contenga todos los datos, ya que la clave no tiene por qué estar formada por todos ellos. Si los valores que forman la clave son vacíos el resultado seria "#".
  - Parámetros
    - FacadeIdentificationStruct: Objeto utilizado para representar un elemento del catálogo.
- getKey: Devuelve la cadena representativa de la clave.
  - Retorno
    - String: cadena representativa de la clave.
- equals: Permite comparar su propia clave con la que recibe como parámetro.
  - Parámetros
    - FacadeKey: clave a comparar.
  - Retorno
    - boolean: Devuelve *verdadero* si son iguales y *falso* en caso contrario.

IFacadeIdentificationRead

Interfaz que debe cumplir un lector del catálogo para poder obtener sus elementos. Va a permitir leer el documento que contiene los datos del catálogo y devolver una lista de todos ellos.

Atributos

No tiene atributos

Métodos

- getDefinitionList: Devuelve la lista de elementos del Catálogo.
  - Retorno
    - ArrayList: Lista con elementos del catálogo.

IFacadeRepresentation

Interfaz de las clases que representan el documento de datos. Cada estándar de documento de perfil de usuario tendrá su propio interfaz que debe heredar de este para que pueda ser tratado por el subsistema.

Atributos

No tiene atributos

Métodos

- setDocument: Método abstracto que permitirá inicializar el documento de datos que va a representar la clase.
  - Parámetros
    - Document: Documento que va a representar la clase.
- getDocument: Método abstracto que permitirá obtener el documento de datos que va representa la clase.
  - Retorno
    - Document: Documento que representa la clase.

IFacadeService

Interfase que deben implementar todos los servicios que puede gestionar el gestor de fachadas y que va a interpretar los estándares de perfiles de usuario.

Esta clase representa la fachada que se proporcional al sistema global para tratar el perfil del usuario.

Toda la información que el sistema necesite del perfil de usuario o la funcionalidad asociada a estos datos debería definirse en este interfaz como un método abstracto para que fuese implementada por cada uno de los servicios interpretes de los estándares.

Atributos

No tiene atributos

Métodos

- setRepresentation: Permite al servicio guardar la referencia a la clase que representa el estándar del documento con los datos del perfil de usuario.
  - Parámetros
    - IFacadeRepresentation: clase que representa el estándar del documento con los datos del perfil de usuario.
- getRepresentation: Permite obtener la clase que representa el estándar del documento con los datos del perfil de usuario.

- Retorno

  - IFacadeRepresentation: clase que representa el estándar del documento con los datos del perfil de usuario.

El resto de métodos que fuesen creados en esta clase, serían métodos abstractos donde la implementación la realizarían los distintos servicios intérpretes de los estándares de perfiles de usuario, como el que a continuación mostramos de ejemplo.

- saveBBDD: Método abstracto que implementarán los servicios de interpretación de estándares que se creen. Permite obtener los datos significativos del perfil de usuario para guardarlos en la estructura de Base de Datos del sistema.

8.5.9 Diagrama de clases de implementación del subsistema de fachadas

El desarrollo central del subsistema de fachadas se completa con la implementación de las interfaces que forman el núcleo del subsistema.

Estas implementaciones se harán considerando que el documento del catálogo y los documentos de los perfiles de usuario que intervienen en el sistema, independientemente del estándar utilizado, tiene representación en formato XML.

Por lo tanto el subsistema quedará completado con:

- La implementación del catálogo de servicios para una representación XML.

- La implementación del lector del documento del catálogo en una representación XML.

- La implementación base de los servicios que interpretan los estándares de perfiles de usuario para una representación XML.

- La implementación base de lectores de los documentos de datos de perfiles de usuario.

En la figura 8.5.9 se muestra el diagrama con el desarrollo central del subsistema, representado en detalle las implementaciones que no fueron mostradas en el diagrama anterior.

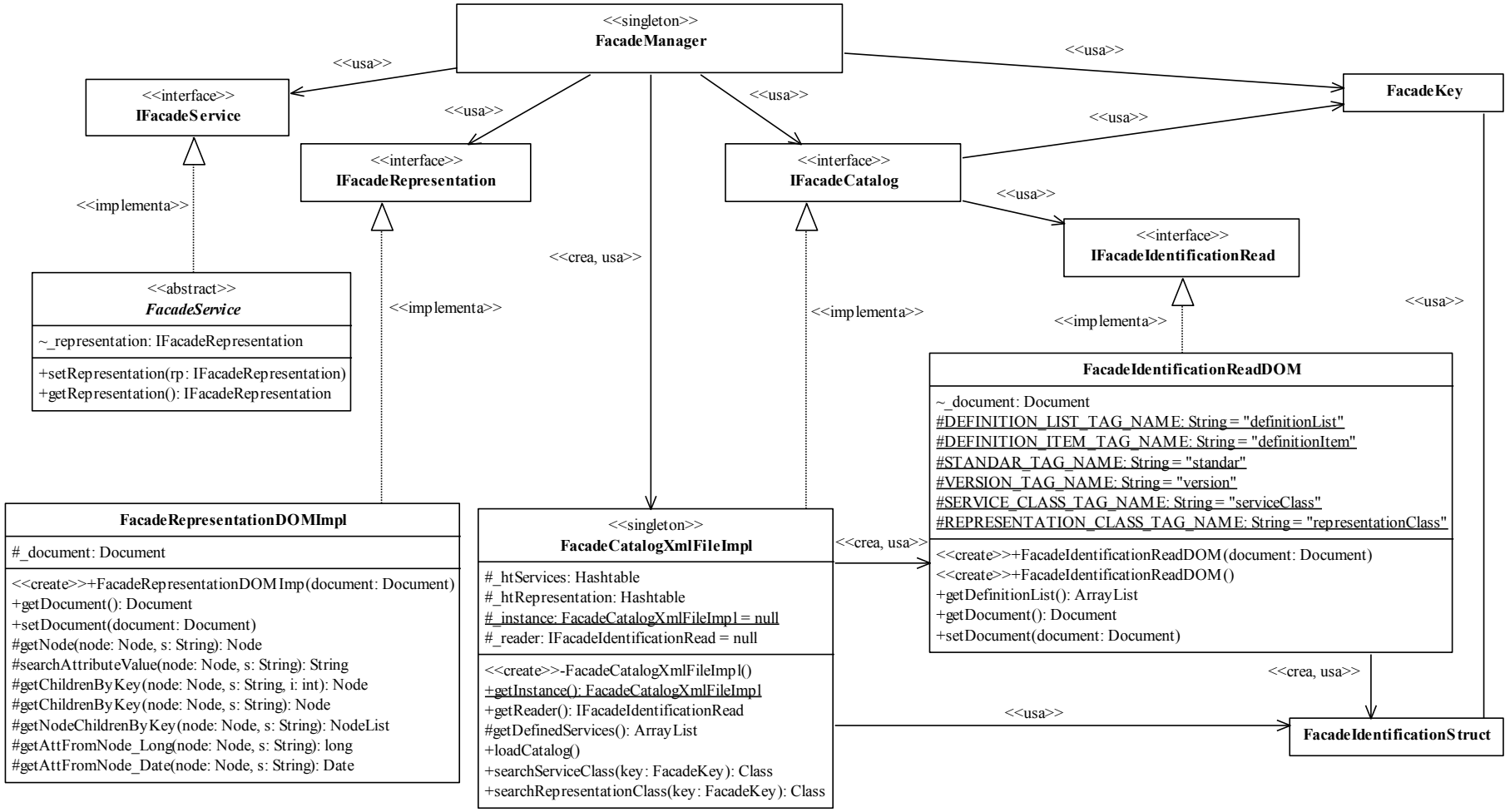

Figura 8.5.9 – Diagrama de Clases de Implementación del Subsistema de Fachadas

La implementación del catálogo debe ser una clase singleton ya que únicamente tiene que existir una instancia en el sistema.

En el diagrama podemos ver que existe una relación entre el gestor de fachadas, *FacadeManger*, y la implementación del catálogo, *FacadeCatalogXMLFileImpl*, ya que el gestor de fachadas es el que va a contener el catálogo y se va a encargar de crearlo y utilizarlo para resolver las peticiones.

La implementación del catálogo es la encargada de crear el lector para leer los elementos del documento y guardarlos en memoria.

La implementación del lector del catálogo, *FacadeIdentificationReadDOM*, tiene una asociación con los elementos del catálogo, *FacadeIdentificationStruct*, ya que se va a encargar de crearlos durante la lectura del documento del catálogo y devolver la lista de todos ellos al catálogo.

Pasamos a describir con más detalle las clases de implementación que intervienen en este diagrama.

FacadeCatalogXmlFileImpl

Representa el catálogo del subsistema de fachadas. Es la implementación del catálogo dentro del subsistema para una representación XML. Esta clase se crea como un singleton que se carga con toda la información del catálogo en el momento de su creación. Proporciona funcionalidad que permite refrescar el catálogo si fuese necesario porque hubiese sufrido algún cambio.

Atributos

- FacadeCatalogXmlFileImpl *_instance:* catálogo del subsistema de fachadas. Es el único objeto que va a ser creado de esta clase.
- IFacadeIdentificationRead *_reader*: lector que va a usar el catálogo para poder leer los elementos de su documento.
- Hashtable *_htServices*: lista de objetos Class correspondientes a las clases servicio y que están descritas en el catálogo.
- Hashtable *_htRepresentation*: Lista de objetos Class descrita en el catálogo, correspondientes a las clases que representan los documentos con los datos en un estándar dado.

Métodos

- FacadeCatalogXmlFileImpl: constructor privado para cumplir el patrón singleton y que así exista únicamente un catálogo. En el momento de ser creado, instanciará el lector del documento y cargará los datos del catálogo.

- getInstance: Crea o devuelve la instancia ya creada del catálogo.
  - Retorno
    - FacadeCatalogXmlFileImpl: instancia del catálogo.
- getReader: Devuelve el lector del catálogo.
  - Retorno:
    - IFacadeIdentificationRead
- getDefinedServices: Se vale del lector del catálogo para obtener todos los elementos declarados en el documento.
  - Retorno: Devuelve la lista de elementos del catálogo que son objetos del tipo FacadeIdentificationStruct.
    - ArrayList: Lista de objetos de tipo FacadeIdentificationStruct.
- searchServiceClass: Devuelve el servicio que interpreta el estándar indicado en la clave que recibe como parámetro. En el caso de no encontrarlo, devolverá una excepción que lo indique. Busca por el FacadeKey en la lista de clases ya cargadas en el Hashtable _htServices la clase que corresponda y la devuelve.
  - Parámetros
    - FacadeKey: Clave del catálogo para la que se desea obtener el servicio.
  - Retorno
    - Class: Clase servicio solicitada
- searchRepresentationClass: Devuelve la clase que representa el documento estándar del perfil de usuario. En caso de no encontrarla, devolverá una excepción que lo indique. Busca por el FacadeKey en la lista de clases ya cargadas en el Hashtable _htRepresentation, la clase que corresponda y la devuelve.
  - Parámetros
    - FacadeKey: Clave del catálogo para la que se desea obtener la representación del documento.
  - Retorno
    - Class: Clase que representa el documento estándar del perfil de usuario.
- loadCatalog: Carga todos los datos del catálogo, si el catálogo ya había sido cargado anteriormente, realizará la recarga. En caso de producirse un error los datos conservarán su estado anterior, por lo que si se había realizado una carga previamente, conservará esos datos.

FacadeIdentificationReadDOM

Implementa la lectura de los elementos del catálogo para una representación XML del mismo. Conoce la representación XML con la que se ha creado el catálogo y permite leerlo generando una estructura java más fácil de utilizar.

Atributos

- Document *_document*: Documento con los datos que del catálogo

El resto de atributos son las constantes que definen el nombre de las etiquetas XML del documento.

- DEFINITION_LIST_TAG_NAME = "definitionList"
- DEFINITION_ITEM_TAG_NAME = "definitionItem"
- STANDAR_TAG_NAME = "standar"
- VERSION_TAG_NAME = "version"
- SERVICE_CLASS_TAG_NAME = "serviceClass"
- REPRESENTATION_CLASS_TAG_NAME = "representationClass"

Métodos

- FacadeIdentificationReadDOM: Constructor sin parámetros.
- FacadeIdentificationReadDOM: Constructor que recibe el documento del catálogo para inicializar el lector.
  - Parámetros
    - Document _document: Catálogo.
- getDocument: Devuelve el documento del Catálogo que contiene el lector.
  - Retorno
    - Document
- setDocument: Inicializa el lector con el documento del Catálogo.
  - Parámetros
    - Document
- getDefinitionList: Obtiene la lista de elementos del catálogo. Cada vez que es llamado se vuelve a generar la lista por si se hubiese modificado el objeto con otro

documento diferente. Por cada elemento del catálogo va a generar una instancia de la clase FacadeIdentificationStruct hasta obtener la lista completa.

- Retorno: Devuelve en un array la lista de elementos del catálogo, o sea, una lista de objetos FacadeIdentificationStruct.
  - ArrayList: lista de objetos FacadeIdentificationStruct.

FacadeService

Clase abstracta base para todos los servicios que interpreten los estándares de perfil de usuario definidos en el subsistema de fachadas. Los únicos métodos que va a implementar del interfaz *IFacadeService* son los que se muestran a continuación.

Atributos

IFacadeRepresentation _representation: Instancia de la clase que representa el documento estándar de perfil de usuario que el servicio sabe interpretar.

Métodos

- setRepresentation: Permite al servicio guardar la referencia a la clase que representa el estándar del documento con los datos del perfil de usuario.
  - Parámetros
    - IFacadeRepresentation: clase que representa el estándar del documento con los datos del perfil de usuario.
- getRepresentation: Permite obtener la clase que representa el estándar del documento con los datos del perfil de usuario.
  - Retorno
    - IFacadeRepresentation: clase que representa el estándar del documento con los datos del perfil de usuario.

FacadeRepresentationDOMImpl

Representa un mecanismo de lectura de los documentos XML de los estándares de perfiles de usuario que el subsistema puede tratar. Proporciona funcionalidad básica para poder leer los elementos de estos documentos. Cada Representación de un estándar de perfil de usuario deberá heredar de esta clase para poder leer de forma sencilla cualquier elemento del documento que representa.

Atributos

- Document *_document*: Documento con los datos del perfil de usuario.

Métodos

- FacadeRepresentationDOMImp: Constructor al que se le indicará el documento con los datos.
  - Parámetros
    - Document
- setDocument: Método abstracto que permitirá inicializar el documento de datos que va a representar la clase.
  - Parámetros
    - Document: Documento que va a representar la clase.
- getDocument: Método abstracto que permitirá obtener el documento de datos que va representa la clase.
  - Retorno
    - Document: Documento que representa la clase.

Esta clase contiene un conjunto de métodos que no vamos a describir, que proporcionan funcionalidad para acceder a los distintos nodos y atributos del documento XML.

8.5.10 Ejemplo de diagrama de clases de implementación con estándares de perfil de usuario

Para comprender mejor el marco de desarrollo que proporciona el subsistema de fachadas, en el siguiente diagrama mostramos cómo estaría el sistema diseñado con un ejemplo de estándares de perfil de usuario.

Vamos a suponer que tenemos dos estándares de perfiles de usuario, estándar uno y estándar dos.

Para cada uno de estos estándares tenemos el documento XML con la definición de todos los datos necesarios para crear un perfil de usuario. La estructura del documento de cada estándar es diferente.

Para poder leer estos documentos vamos a generar, por cada estándar, un interfaz y una implementación basada en la lectura de un documento XML, que es lo que llamamos *Representación* del estándar de un documento de perfil de usuario.

Por lo tanto las clases resultantes serian:

- Interfaz IStandarOneRepresentation para el estándar uno.

- Clase StandarOneRepresentationDOMImpl para el estándar uno.

- Interfaz IStandarTwoRepresentation para el estándar dos.

- Clase StandarTwoRepresentationDOMImpl para el estándar dos.

Los dos interfaces van a heredar de la interfaz *IFacadeRepresentation* y las dos clases heredan de *FacadeRepresentation* para cumplir con el marco de desarrollo definido en el subsistema de Fachadas.

El siguiente paso es generar los servicios que interpretan cada uno de estos estándares. Por lo tanto se crearan las siguientes clases:

- StandarOneService para el estándar uno.

- StandarTwoService para el estándar dos.

Cada uno de estos servicios es la fachada que el subsistema proporciona al sistema global y es necesario que hereden de la clase *FacadeService*.

En estos servicios se realizará la implementación de los métodos que obtienen toda la información del perfil de usuario necesaria para el sistema. La implementación en cada servicio será diferente ya que dependerá de cómo este representada por cada estándar.

Todos los métodos que el sistema va a necesitar se definirán en el interfaz *IFacadeService* como métodos abstractos, donde como se ha comentado anteriormente la implementación se realizará en el servicio de cada estándar.

De esta manera conseguimos proporcionar una única fachada al sistema y le evitamos tener que trabajar directamente con cada uno de los servicios, consiguiendo un bajo acoplamiento entre el sistema y los estándares de usuario.

En la figura 8.5.10a se puede ver un diagrama del subsistema implementado con unos estándares generales de ejemplo.

–

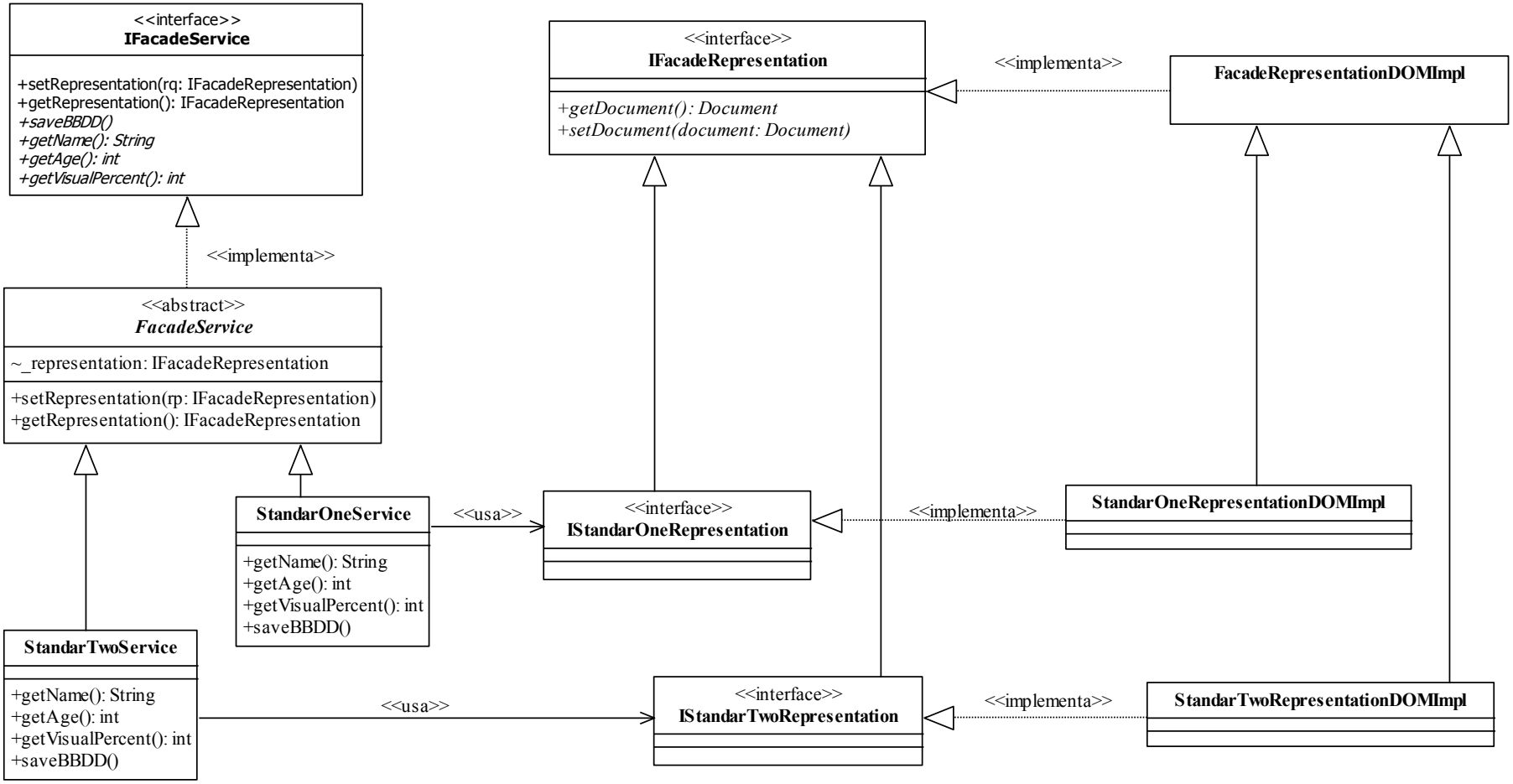

Figura 8.5.10a – Diagrama de clases de Ejemplo de Implementación con Estándares

En el diagrama se ve representado un ejemplo de la necesidad del sistema por obtener datos del usuario como el nombre, edad o porcentaje de visión de la que dispone, además de funcionalidad para grabar los datos relevantes para el sistema en Base de Datos.

Estos métodos se han incluido en el interfaz *IFacadeService*, aunque podrían existir otras soluciones que evitasen tocar las clases pertenecientes al marco de desarrollo del subsistema de fachadas como por ejemplo, crear una clase interfaz genérica que herede del *IFacadeService* donde se incluyan todos los métodos abstractos necesarios y que los servicios creados para los estándares, aparte de heredar de la clase *FacadeService* implementen el interfaz genérico.

El sistema lo único que debería es transformar el servicio que devuelve el subsistema como un *IFacadeService* al interfaz genérico *IGenericStandarInterface*.

En la Figura 8.5.10b se muestra el diagrama de clases representativo para este interfaz genérico.

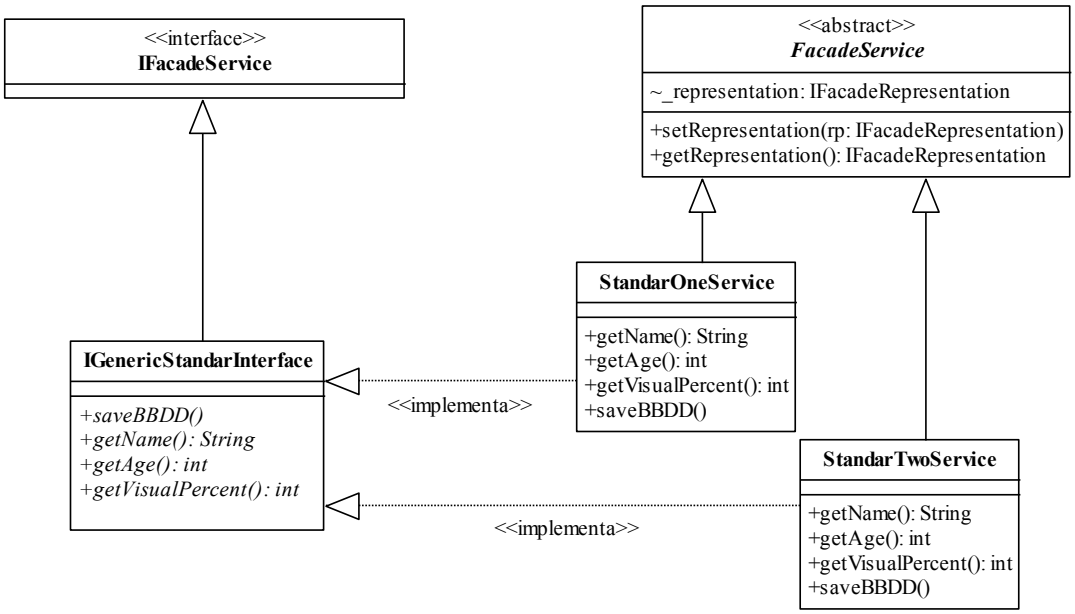

Figura 8.5.10b - Interfaz genérico para los servicios intérpretes de estándares

Otra de las soluciones sería que el sistema únicamente tratase con el gestor de fachadas, sin necesidad de que este le proporcionase el interfaz para resolver sus peticiones.

El sistema realizaría todas las peticiones de datos que desea obtener del usuario al gestor de fachadas, lo que supone que este por cada petición se encargase de resolver las clases que resuelven el estándar y la gestión de la petición.

El único cambio respecto a la solución de la figura 8.4.9b es que el gestor de fachadas implementaría también todos los métodos del interfaz IGenericStandarInterface y para cada uno de ellos resolvería las clases que interpretan.

De esta manera se cumpliría el patrón Facade fielmente, pero teniendo en cuenta que solo existe una instancia del gestor de fachadas en el sistema y que debería encargarse de resolver cada una de las peticiones, esta opción no sería nada óptima.

Por lo tanto la mejor implementación es proporcionar la Fachada al sistema para que trabaje con ella.

8.5.11 Diagramas de secuencia

En este apartado del proyecto vamos a mostrar el funcionamiento dinámico del subsistema de fachadas mediante unos diagramas de secuencia.

Dentro de los posibles escenarios que se plantean en el subsistema nos vamos a centrar en los más importantes o relevantes como son:

- La inicialización del subsistema.
- La carga/recarga del catálogo.
- La obtención del servicio intérprete de un estándar de perfil de usuario.

8.5.12 Diagrama de secuencia de inicialización del subsistema

En este diagrama se muestra la creación inicial del gestor de fachadas y del catálogo. La inicialización solo se realiza una vez, cuando a la clase *FacadeManager* se le solicita su instancia.

Al tratarse de una clase singleton, únicamente la primera vez creará la instancia y cargará el catálogo por lo que en sucesivas llamadas únicamente será necesario devolver la instancia que ya fue creada.

En la figura 8.5.12 se muestra el diagrama de secuencia de inicialización.

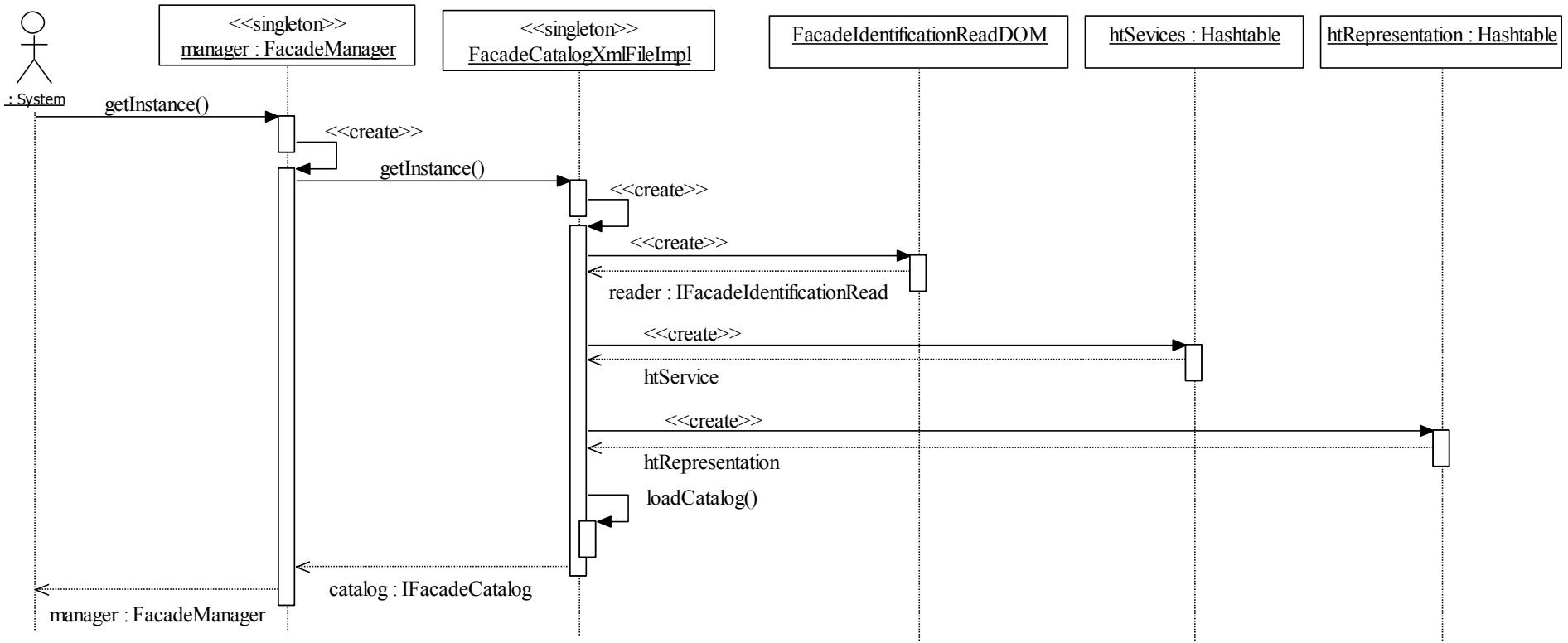

Figura 8.5.12 - Diagrama de Secuencia de Inicialización del Subsistema de Fachadas

Como se puede ver en el diagrama, el flujo del proceso comienza cuando el sistema, que en nuestro caso sería el módulo de Personalización de Contenido, solicita la instancia del Gestor de Fachadas (*FacadeManager*). Esto va a provocar una serie de acciones internas, totalmente transparentes al actor, que permiten la inicialización del subsistema de fachadas.

El *FacadeManager* crea su instancia y solicita la instancia del catálogo. En este caso la solicitud la realiza directamente sobre la implementación XML del catálogo (*FacadeCatalogXMLFileImpl*).

El catálogo está implementado con un singleton por lo que únicamente existirá una instancia en el subsistema. Con esta primera llamada se crea la instancia del catálogo, el cual va a inicializarse.

Se crea el lector que le va a permitir interpretar el documento XML con la definición del catálogo. Ya que el catálogo está implementado para un formato XML, el lector que necesita es el *FacadeIdentificationReadDOM*.

Se crea un *Hashtable* de Servicios en el que guardará las clases *Service* declaradas en el catálogo.

Se crea un *Hashtable* de Representaciones en el que guardará las clases *Representation* declaradas en el catálogo.

Una vez inicializado, el catálogo se dispone a cargar los datos mediante la llamada al método *reloadServices*, cuyo funcionamiento se puede ver con más detalle en el diagrama de secuencia de carga del catálogo representado en la figura 8.4.12.

8.5.13 Diagrama de secuencia de carga del catálogo

Este diagrama representa como el catálogo es capaz de cargarse/recargase a petición del gestor de fachadas, lo que le permitiría estar siempre actualizado.

En la figura 8.5.13 se muestra el diagrama de secuencia de la carga de datos.

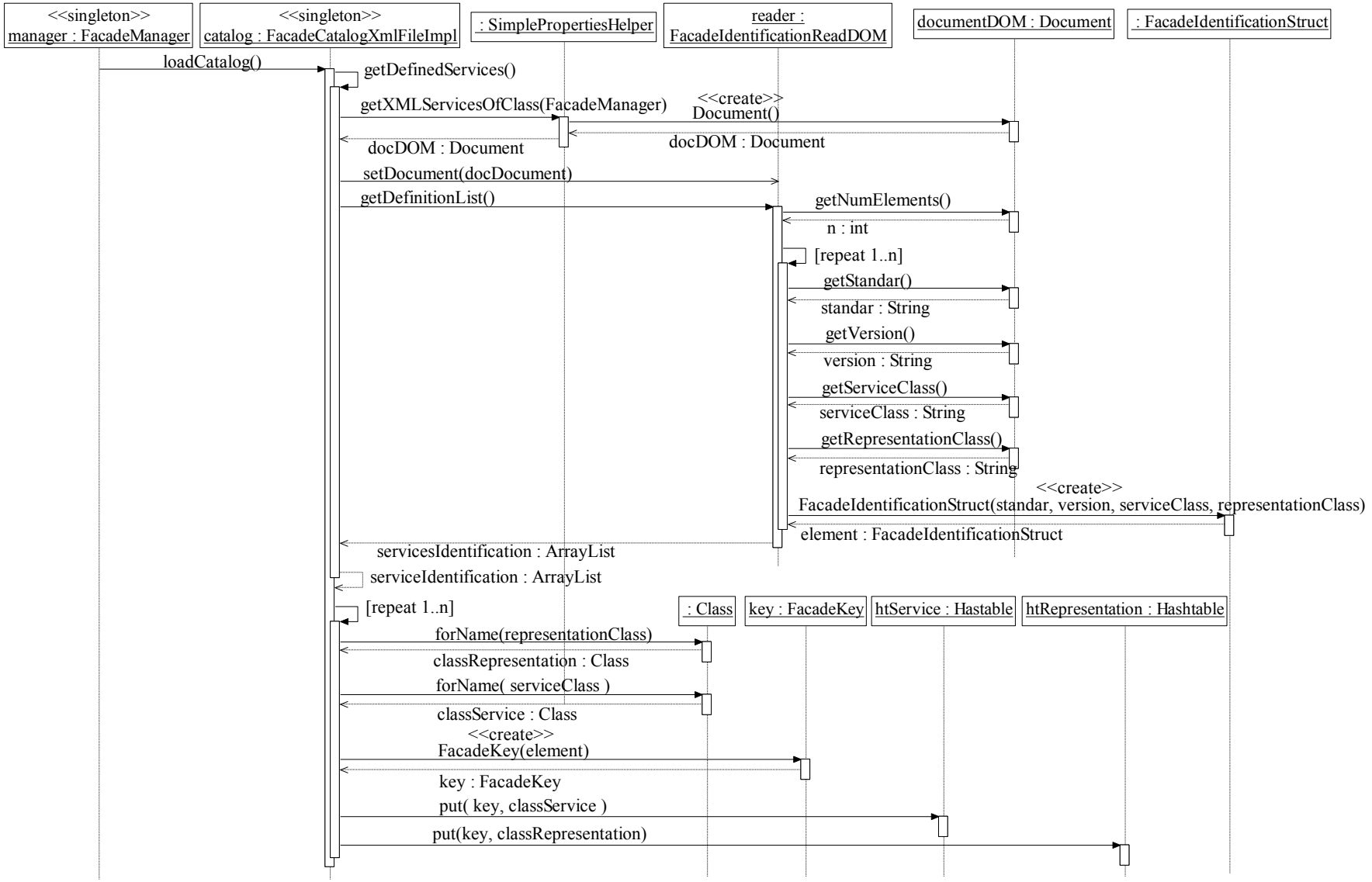

Figura 8.5.13 – Diagrama de Secuencia de Carga del Catálogo

Como se puede ver en el diagrama, inicialmente el *manager* realiza la solicitud al catálogo que ya tiene instanciado (*catalog*), para que realice la carga mediante el método *loadCatalog*.

La carga del catálogo estaría dividida en dos partes:

1. Extraer la declaración del catálogo de un documento XML.

2. Creación de las clases declaradas en el catálogo y almacenamiento para su tratamiento posterior.

Pasamos a explicar cada una de las partes.

1. Extraer la declaración del catálogo de un documento XML

El catálogo va a leer el documento XML donde se encuentra la declaración del mismo, o sea, los nombres de las clases *Representación* y *Servicio* que se utilizan para interpretar un estándar de perfil de usuario. El método encargado de realizar esta tarea es *getDefinedServices*.

El catálogo solicita al *SimplePropertiesHelper* la representación en árbol DOM (clase *Document* en java) del documento XML, con la declaración del catálogo.

El siguiente paso del catálogo es inicializar su lector de documentos XML (*reader*) con este *Document* y solicitarle que la lista de identificaciones declaradas, mediante el método *getDefinitionList*.

El lector *reader* se encarga de leer los atributos de cada línea del documento (standar, version, serviceClass, representationClass) para crear cada elemento de identificación del catálogo (*FacadeIdentificationStruct*) que va guardando en un array.

El catálogo guarda el array recibido del *reader* con las declaraciones.

2. Creación de las clases declaradas en el catalogo y almacenamiento para su tratamiento posterior

El catálogo va ser capaz de interpretar la declaración de clases para crear los elementos y almacenarlos de forma que sea sencillo su acceso cuando necesite atender una petición.

Va a crear la clase java declarada como *RepresentationClass*. Esta clase deberá existir dentro del paquete java.

Va a crear la clase java declarada como *ServiceClass*. Esta clase deberá existir dentro del paquete java.

Va a crear una clase clave *FacadeKey* para identificar estos elementos. Esta clave está compuesta por el estándar y la versión.

Estos datos los va a guardar por su clave de identificación en dos Hashtable, uno con las clases servicio y otro con las clases representación creadas.

8.5.14 Diagrama de secuencia se solicitud de un servicio

En el siguiente diagrama vamos a ver cómo se comporta el marco de desarrollo del subsistema de fachadas ante una llamada para obtener el servicio que permite interpretar el perfil de usuario que interactúa con el sistema, de esta forma podremos llegar a comprender mejor su funcionamiento.

En la figura 8.5.14 se muestra el diagrama de secuencia para solicitar un servicio al subsistema de fachadas.

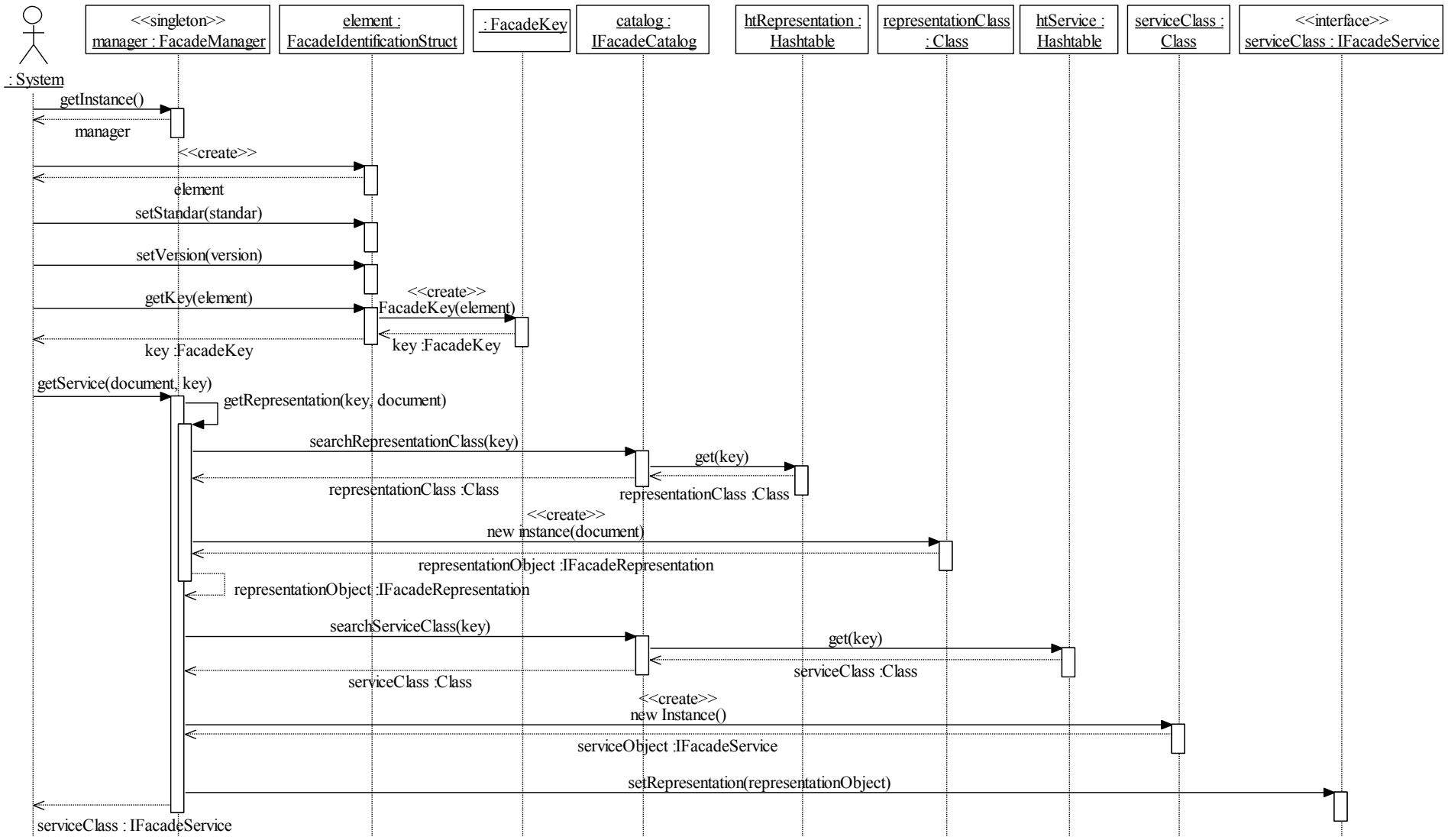

Figura 8.5.14 – Diagrama de Secuencia de Petición de Servicio

Como se puede ver en el diagrama, el sistema necesita primeramente obtener el gestor de fachadas *manager* para poder realizar la solicitud del servicio.

La solicitud se realiza mediante el método *getService*, pero es necesario indicarle dos parámetros que son la clave y el documento con los datos del perfil de usuario.

Para obtener la clave el sistema tendrá identificado el estándar con el que está definido el perfil del usuario que interactúa en el sistema EU4ALL, mediante los valores e*stándar* y *versión*. Con estos dos valores se va a crear un elemento de identificación *FacadeIdentificactionStruct* a través del cual va a obtener la clave mediante el método *getKey.*

Seguidamente con la clave y el documento de los datos del perfil de usuario el sistema va a solicitar al gestor de fachadas el servicio adecuado que realizará las acciones siguientes.

Lo primero es conseguir la representación del estándar. El gestor de fachadas va a obtener la clase adecuada que representa los datos del documento del perfil de usuario. Ésta se la solicitará al catálogo mediante el método *searchRepresentationClass*, al que únicamente necesitará indicarle la clave.

El catálogo buscará en su lista de clases representación la identificada por la clave y se la devolverá al gestor de fachadas.

El siguiente paso del gestor de fachadas será crear la instancia de la clase representación inicializándola con el documento de datos del perfil de usuario.

Una vez obtenida la representación adecuada, el gestor de fachadas debe obtener el servicio por lo que solicita al catálogo la clase servicio. Está solicitud se realiza mediante el método *searchServiceClass*, indicándole la clave.

El catálogo buscará en la lista de clases servicio la identificada por la clave y se la devolverá al gestor de fachadas.

El gestor de fachadas creará la instancia de la clase servicio y lo inicializará con la clase representación obtenida anteriormente.

Finalmente el gestor de fachadas va a devolver al sistema el servicio capaz de interpretar los datos del perfil de usuario que interactúa con el sistema que además contendrá la clase que representa el documento con los datos y en la que se apoyará para resolver las peticiones.

8.6 Pruebas Unitarias

En las siguientes tablas se ven representadas las pruebas unitarias realizadas y los resultados obtenidos.

Las pruebas unitarias realizadas van encaminadas a la validación de la funcionalidad propuesta así como la respuesta ante fallos.

Pruebas unitarias de Inicialización del subsistema:

1	Caso de Uso: Inicialización del Subsistema				
Nº	Caso de prueba	Condiciones	Datos de prueba	Resultado Esperado	Evaluación
1.1	Obtener gestor de fachadas	Subsistema no inicializado	Documento del catálogo	Instancia del gestor de fachadas	Correcto. El catálogo ha sido cargado y creado el gestor de fachadas devuelto a la petición
1.2	Obtener gestor de fachadas inicializado	Subsistema ya inicializado	Documento del catálogo	Instancia del gestor de fachadas	Correcto. Se obtiene la instancia del gestor de fachadas que ya se había creado anteriormente
1.3	Control de error de inicialización	Subsistema no inicializado	Documento de catalogo con definición errónea	Sistema no inicializado y retorno del error	Correcto. No se realiza inicialización y se devuelve el error controlado

Pruebas unitarias de Carga del catálogo:

2	Caso de Uso: Carga del Catálogo de Servicios				
Nº	Caso de prueba	Condiciones	Datos de prueba	Resultado Esperado	Evaluación
2.1	Carga/recarga del Catálogo		Documento del catálogo	Catálogo cargado con la definición	Correcto. El catálogo se queda cargado con los datos del documento
2.2	Incorporación de servicio al catálogo	Catálogo ya cargado	Documento del catálogo con la nueva definición incorporada	Catálogo cargado con la nueva definición	Correcto. El catálogo ha sido actualizado con el nuevo servicio
2.3	Validación de clase servicio		Documento del catálogo con definición de servicio inexistente	Retorno del error	Correcto. Se devuelve el error y si el catálogo estaba cargado anteriormente conserva sus datos

2.4	Validación de clase Representación		Documento del catálogo con definición de servicio inexistente	Retorno del error	Correcto. Se devuelve el error y si el catálogo estaba cargado anteriormente conserva sus datos

Pruebas unitarias de Petición de servicio al subsistema:

3	Caso de Uso: Petición de Servicio				
Nº	**Caso de prueba**	**Condiciones**	**Datos de prueba**	**Resultado Esperado**	**Evaluación**
3.1	Obtener servicio del catálogo	Subsistema inicializado	Documento de datos del usuario. Nombre de estándar:AccLip Versión:1	Interfaz IFacadeService	Correcto. Se recupera el servicio AccLipService a través del interfaz.
3.2	Recuperar dato visualPercent	Subsistema ya inicializado	Documento de datos del usuario. Nombre de estándar:AccLip Versión:1	Valor entero del dato	Correcto. Se obtiene el interfaz fachada y se accede al dato
3.3	Recuperar dato visualPercent	Subsistema ya inicializado	Documento de datos del usuario. Nombre de estándar:AccesForAll Versión:1	Valor entero del dato	Correcto. Se obtiene el interfaz fachada y se accede al dato
3.4	Error de identificación	Subsistema ya inicializado	Documento de datos del usuario:AccesForAll) . Nombre de estándar:AccLip Versión:1	Retorno del error	Correcto. Se devuelve error controlado indicando fallo de representación.

Las pruebas unitarias realizadas y la preparación de los datos de prueba se han realizado en el entorno de desarrollo del subsistema, sobre un ordenador personal con S.O. Windows XP.

Para poder realizar las pruebas ha sido necesaria la generación de las clases Representación, que interpretaban los documentos XML [19] [21] con la definición de los estándares AccLIP y AccessForAll. Estas clases junto con los servicios han sido proporcionadas como parte del subsistema al sistema global para su integración.

9 Conclusiones y Líneas futuras de trabajo

A continuación presentaremos las conclusiones obtenidas sobre el trabajo realizado así como conclusiones a nivel personal y mostraremos líneas futuras que permitan una continuidad de este proyecto fin de carrera.

9.1 Conclusiones de trabajo

Con la realización de este proyecto fin de carrera cubriríamos ciertos aspectos que indicamos a continuación.

Hemos implementado un subsistema de fachadas que resuelve la falta de unificación de estándares de perfiles de usuarios.

Permite abstraer las especificaciones utilizadas en cada estándar de perfil de usuario proporcionando una única fachada con la que trabajar.

Se ha realizado con un diseño genérico que le va a permitir crecer, cambiar y reutilizarse fácilmente. Gracias a este diseño, con pequeños cambios, se podrán cubrir otras necesidades funcionales distintas al propósito para el que se realizó la implementación, orientada a interpretar estándares de perfil de usuario.

El subsistema de fachadas constituye un componente del Módulo de Personalización de Contenido, que le va a aislar de la tarea de interpretar la definición en la que se encuentre la información de los usuarios. Hemos visto que el subsistema lo podríamos ubicar en distintas partes del módulo de personalización para interpretar características de dispositivos y de objetos de aprendizaje.

Hemos mostrado las características principales del proyecto europeo EU4ALL, que es el marco de trabajo en el que se engloba el PFC y cuyo objetivo principal es mejorar la eficacia y eficiencia de estrategias de aprendizaje para personas con discapacidad

Podríamos concluir que este proyecto fin de carrera suma su aportación para conseguir que el acceso al aprendizaje llegue a todos.

9.2 Conclusiones personales

Ha sido gratificante poder colaborar en un proyecto europeo donde participan diversidad de empresas, organismos y universidades con un objetivo global enfocado a ayudar a la sociedad.

Supone una satisfacción personal poner un granito de arena para avanzar en el campo de la accesibilidad, que va a permitir que personas que se encuentran en condiciones diferentes, no vean limitado su acceso al aprendizaje o a cualquier tipo de información a través de las actuales y futuras tecnologías.

9.3 Líneas futuras

El marco de desarrollo diseñado proporciona el mecanismo para interpretar distintos estándares de usuario con la capacidad de incorporar fácilmente implementaciones para interpretar nuevos estándares o versiones emergentes.

Éste diseño propone las interfaces (IFacadeService e IFacadeRepresentation) que deben cumplir los intérpretes de los documentos de los estándares, pero resulta necesaria la creación de sus implementaciones. En el proyecto esta tarea se ha realizado en su mayoría de forma manual.

Una línea de trabajo futura de este proyecto sería proporcionar un mecanismo o herramienta que permita la generación automática de estas clases a partir del documento de definición del estándar. De ésta manera la incorporación de nuevos estándares o versiones de estándar a interpretar por el sistema, sería automática.

Ésta herramienta debería permitir analizar los documentos XML de definición de los estándares y crear las clases necesarias para su interpretación, adaptadas al marco de trabajo del software implementado. A su vez la herramienta debería generar la nueva entrada a añadir en la definición del catálogo.

10 Manual de usuario

Este manual va a permitir al resto de módulos que vayan a utilizar el subsistema de fachadas conocer las particularidades de configuración, representación de datos y uso del subsistema.

Va a mostrar cómo utilizar el subsistema tal y como está implementado para interpretar los estándares de modelado de perfiles de usuario.

Intentará indicar como se puede adaptar fácilmente el subsistema a las necesidades propias de cualquier módulo que requiera de su utilización.

10.1 Tecnología

A nivel tecnológico el marco de desarrollo del proyecto lo compone:

- Java JDK 1.5 [16]: El desarrollo del subsistema se realiza en Java.
- JAXP 1.2: API Java para procesamiento de XML. [22] [23]
- Sistema Operativo: Windows XP

El *software* utilizado para el desarrollo del proyecto se detalla a continuación:

- StarUML 5.0 [18]: se utiliza esta herramienta de libre distribución para el diseño de software con notación UML.

  Web: http://staruml.sourceforge.net/en/

- Eclipse 3.3 [17]: se utiliza esta herramienta de libre distribución para la codificación del software en el lenguaje java.

  Web: http://www.eclipse.org/

- XMLPad 3.0: se utiliza esta herramienta de libre distribución para la creación y edición de documentos XML.

  Web: http://www.wmhelp.com/

- Microsoft Project 2000: utilizado para realizar la planificación del proyecto.
- Microsoft Office 2010: utilizado para la generación de documentación.

10.2 Paquetes

Vamos a ver la estructura de paquetes que se ha utilizado para la implementación, que viene dada por el diseño del proyecto EU4ALL, por lo que el subsistema de fachadas debe incluirse dentro del paquete de componentes de éste proyecto (*isl.einclusion.eu4all*)

El subsistema de fachadas se ha desarrollado para la interpretación de perfiles del modelo de usuario dentro de la personalización de contenido, por lo que las clases se encuentran ubicadas en el paquete de modelo de usuario dentro del paquete de personalización de contenido (*isl.einclusion.eu4all.cp.um*) dónde,

- cp significa *Content Personalisation*
- um significa *User Model*

Para que se encuentre bien diferenciada la ubicación del subsistema de fachadas se ha creado un paquete *isl.einclusion.eu4all.cp.um.facade* donde se incluye todo el software desarrollado.

A continuación mostramos un diagrama de paquetes con la distribución de las clases e interfases que componen el subsistema de fachadas.

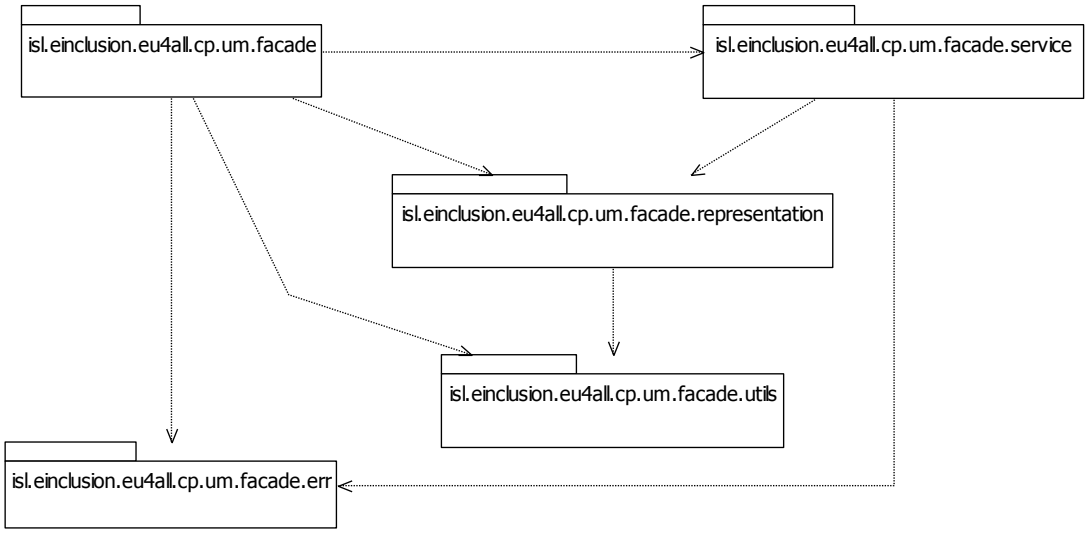

Figura 10 – Diagrama de Paquetes del Subsistema de Fachadas

El desarrollo del subsistema de fachadas lo hemos situado en el paquete *facade* dentro del paquete de Modelo de Usuario (um) que pertenece al de Personalización de Contenido (cp).

En el paquete *facade* se encuentra el desarrollo del gestor de fachadas y el catálogo y se separan en subpaquetes los servicios, representaciones, utilidades para y gestión de errores.

Dentro de cada uno de estos paquetes vamos a indicar el conjunto de clases que lo forman.

Paquete del núcleo del subsistema

isl.einclusion.eu4all.cp.um.facade

Las clases incluidas en este paquete son:

- *FacadeManager*
- *FacadeKey*
- *IFacadeCatalog*
- *IFacadeIdentificationRead*
- *FacadeIdentificationStruct*
- *FacadeCatalogXmlFileImpl*
- *FacadeIdentificationReadDOM*

Paquete para las representaciones

isl.einclusion.eu4all.cp.um.facade.representation

Las clases incluidas en este paquete son:

- *IFacadeRepresentation*
- *FacadeRepresentationDOMImp*
- Cada representación implementada específica de un estándar

Paquete para los servicios

isl.einclusion.eu4all.cp.um.facade.service

Las clases incluidas en este paquete son:

- *IFacadeService*
- *FacadeService*

- Cada servicio implementado específico de un estándar

Paquete de Excepciones

isl.einclusion.eu4all.cp.um.facade.err

Las clases incluidas en este paquete son:

- *PathNotFoundException*
- *ServiceNotFoundException*
- *FacadeProcessException*
- *NotValidRepresentationException*

Paquete de Utilidades

isl.einclusion.eu4all.cp.um.facade.utils

Las clases incluidas en este paquete son:

- *DOMHelper*
- *XMLHelper*

10.3 Documento del catálogo

Para la representación de los datos del catálogo se ha utilizado un documento XML, de ahí que el lector utilizado sea orientado a este formato.

A continuación mostramos un ejemplo del documento XML de declaración del catálogo, para la implementación usada de estándares de modelado de perfiles de usuario.

serviceDefinition.xml

```
<?xml version="1.0"?>
- <main>
  - <definitionList>
    - <definitionItem>
        <standar>AccLip</standar>
        <version>1</version>
        <serviceClass>isl.einclusion.eu4all.cp.um.facade.service.AccLipService</serviceClass>
        <representationClass>isl.einclusion.eu4all.cp.um.facade.representation.AccLipRepresentationDOMImpl</representationClass>
      </definitionItem>
    - <definitionItem>
        <standar>AccessForAll</standar>
        <version>1</version>
        <serviceClass>isl.einclusion.eu4all.cp.um.facade.service.AccessForAllService</serviceClass>
        <representationClass>isl.einclusion.eu4all.cp.um.facade.representation.AccessForAllRepresentationDOMImpl</representationClass>
      </definitionItem>
    </definitionList>
</main>
```

Figura 10.3 – Ejemplo de declaración del catálogo

Este documento de declaración del catálogo iría en el paquete *isl.einclusion.eu4all.cp.um.facade* con el nombre de *servicesDefinition.xml*. El gestor del sistema podrá modificarlo para resolver cualquier error de declaración o incorporar nuevas implementaciones de estándares de perfiles de usuario.

También podría pensarse en declarar los datos del catálogo en Base de Datos y adaptar el subsistema para que fuesen leídos. Únicamente habría que desarrollar un lector que pudiese obtener los datos de la BBDD y un catálogo que utilizase éste lector. Las clases a desarrollar serían por lo tanto:

- *FacadeIdentificationReadBBDD*: nuevo lector que implementaría el interfaz *IFacadeIdentificationRead*

- *FacadeCatalogBBDDImpl*: nuevo catálogo que implementaría el interfaz *IFacadeCatalog* y utilizaría el nuevo lector *FacadeIdentificationReadBBDD*

10.4 Documento de Datos

El documento con los datos del usuario va a venir en formato XML, cuya estructura va a variar dependiendo del estándar de modelado de perfiles de usuario que se utilice.

Debemos partir de la estructura definida en el estándar para generar la clase java que llamamos Representación que nos permite acceder a los datos de todas las etiquetas XML del documento.

En la siguiente figura se puede ver un ejemplo de la estructura XML de datos del estándar *AccLip*, dónde se han descrito algunas de sus características.

```
<?xml version="1.0" encoding="UTF-8"?>
- <tns:accessForAll schemaVersion="0.0.0" xsi:schemaLocation="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:imslip="http://www.imsproject.org/xsd/imsliprootv1p0" xmlns:tns="http://www.imsglobal.org/xsd/accessforallv1p0">
  - <tns:context lang="" external="http://www.altova.com" identifier="ID000000">
    - <tns:display>
        <tns:screenReader> </tns:screenReader>
      - <tns:screenEnhance>
        - <tns:screenEnhanceGeneric>
            <tns:fontFace> </tns:fontFace>
            <tns:fontSize value="2" usage=""/>
            <tns:foregroundColor value="ff000000" usage=""> </tns:foregroundColor>
            <tns:backgroundColor value="ffffffff" usage=""> </tns:backgroundColor>
            <tns:highlightColor value="ffff0000" usage=""> </tns:highlightColor>
            <tns:invertedColorChoice value="false" usage=""> </tns:invertedColorChoice>
            <tns:cursorSize value="0.5" usage=""> </tns:cursorSize>
            <tns:cursorColor value="ffffffff" usage=""> </tns:cursorColor>
            <tns:cursorTrails value="0.5" usage=""> Efecto al mover el raton </tns:cursorTrails>
            <tns:tracking> </tns:tracking>
            <tns:magnification value="" usage=""/>
          </tns:screenEnhanceGeneric>
        </tns:screenEnhance>
        <tns:braille> </tns:braille>
        <tns:tactile> </tns:tactile>
      - <tns:visualAlert>
          alternativas visuales a alertas de sonido.
        - <tns:visualAlertGeneric>
            <tns:systemSounds value="none" usage="">window </tns:systemSounds>
            <tns:captions value="false" usage="">titulos para sonidos del sistema </tns:captions>
          </tns:visualAlertGeneric>
        </tns:visualAlert>
      - <tns:structuralPresentation>
          <tns:contentDensity value="detailed" usage="">cuanto detalle a mostrar </tns:contentDensity>
          <tns:contentViews value="imageIntensive" usage="">Mostrar imagen o texto </tns:contentViews>
          <tns:showLinks value="false" usage="">mostrar lista de links </tns:showLinks>
          <tns:showTranscript value="false" usage="">transcripcion de audio cuando sea posible </tns:showTranscript>
          <tns:showNotes value="true" usage="">mostrar anotaciones cuando sea posible </tns:showNotes>
          <tns:windowLayout value="frontMost" usage="">como mostrar si hay varias ventanas </tns:windowLayout>
        </tns:structuralPresentation>
      </tns:display>
    - <tns:control>
        <tns:keyboardEnhanced> </tns:keyboardEnhanced>
        <tns:onscreenKeyboard> </tns:onscreenKeyboard>
        <tns:alternativeKeyboard> </tns:alternativeKeyboard>
        <tns:mouseEmulation> </tns:mouseEmulation>
        <tns:alternativePointing> </tns:alternativePointing>
        <tns:voiceRecognition> </tns:voiceRecognition>
        <tns:codedInput> </tns:codedInput>
        <tns:structuralNavigation> </tns:structuralNavigation>
      </tns:control>
    - <tns:content>
      - <tns:alternativesToVisual>
          <tns:audioDescription lang="" usage="" type=""/>
          <tns:altTextLang lang="" usage=""/>
          <tns:longDescriptionLang lang="" usage=""/>
          <tns:colorAvoidance/>
        </tns:alternativesToVisual>
      - <tns:alternativesToText>
          <tns:graphicAlternative value="" usage=""/>
          <tns:signLanguage value="" usage=""/>
        </tns:alternativesToText>
        <tns:alternativesToAuditory> </tns:alternativesToAuditory>
        <tns:learnerScaffold value="" usage=""/>
        <tns:personalStylesheet value="" usage=""/>
        <tns:extraTime value="" usage=""/>
      </tns:content>
    </tns:context>
</tns:accessForAll>
```

Figura 10.4 – Ejemplo de estructura XML del estándar AccLip

En el XML algunas de las características se han mostrado con más detalle y se ha indicado una breve descripción de su utilidad para poder entender la estructura.

10.5 Ejemplo de utilización del subsistema

En el siguiente código java de ejemplo vamos a ver como el servicio de personalización de contenido puede obtener el servicio apropiado para interpretar el estándar de perfil de usuario utilizado y así recuperar sus características.

Es necesario que el módulo de personalización de contenido ya tenga cargadas tres variables:

- *doc*: instancia de la clase *Document* con las características del usuario.
- *standar*: identificación del estándar utilizado.
- *version*: versión del estándar para obtener el servicio correspondiente.

Lo primero es obtener la instancia del gestor de fachadas, y el identificador del elemento del catálogo.

```
//Obtenemos la instancia del Gestor de Fachadas
FacadeManager oManager = FacadeManager.getInstance();
//Creamos el elemento identificador del catálogo
FacadeIdentificationStruct oElement = new FacadeIdentificationStruct();
oElement.setStandar(standar);
oElement.setVersion(version);
```

El siguiente paso es obtener el servicio con el que se va a poder acceder a las características del usuario.

```
//Obtenemos el servicio apropiado
IFacadeService oService = null;
oService = oManager.getService(oElement.getKey(), doc);
```

Una vez que tenemos el servicio podemos obtener las características del usuario que se necesiten, por ejemplo el grado de discapacidad visual. Este método accederá a través de la clase representación hasta el atributo XML con el valor buscado.

```
//Recuperamos el grado de vision del usuario
int iVisualPercent = oService.getVisualPercent();
```

11 Bibliografía y Referencias

- [1]Patrones de diseño aplicados a Java. *Stephen Stelting, Olav Maassen.* Editorial Pearson/Prentice Hall
- [2]*EU4ALL WP3.3 Content Personalisation, D3.3.1a State-of-the-art analysis of Content Personalisation*
- [3]*EU4ALL WP3.3 Content Personalisation, D3.3.1b Functional specification for the Content Personalisation Software Module*
- [4]*EU4ALL WP3.3 Content Personalisation, D3.3.2 Integration of SP4 result on content metadata*
- [5]*EU4ALL WP2.2 Open Architecture Design, D2.2.2 General Architecture Design*
- [6]*EU4ALL Annex I – Description of Work*
- [7]http://www.imsglobal.org/profiles/index.html
- [8]http://www.imsglobal.org/metadata
- [9]http://www.imsglobal.org/accessibility
- [10]http://dublincore.org
- [11]http://www.w3.org/TR/2006/WD-CCPP-struct-vocab2-20061208/
- [12]http://www.holub.com/goodies/uml/
- [13]http://edn.embarcadero.com/article/31863
- [14]http://www.scribd.com/doc/399157/UML-20-Cheatsheet
- [15]http://www.eu4all-project.eu
- [16]http://java.sun.com/
- [17]http://www.eclipse.org/
- [18]http://staruml.sourceforge.net/en/
- [19]http://www.wmhelp.com
- [20]http://es.wikipedia.org
- [21]*XML in a Nutshell, second edition. Elliotte Rusty Harold & W. Scott Means. Published by O'Reilly & Associates*

- [22]http://java.sun.com/webservices/reference/tutorials/jaxp/html/intro.html
- [23]http://java.sun.com/xml/downloads/jaxp.html
- [24]http://office.microsoft.com/
- [25]http://www.ibm.com/software/rational/rup/

12 APÉNDICE

12.1 Planificación y Presupuesto del proyecto.

12.1.1 Planificación

Mediante el siguiente diagrama de Gantt se puede observar la planificación realizada para llevar a cabo cada una de las tareas del proyecto.

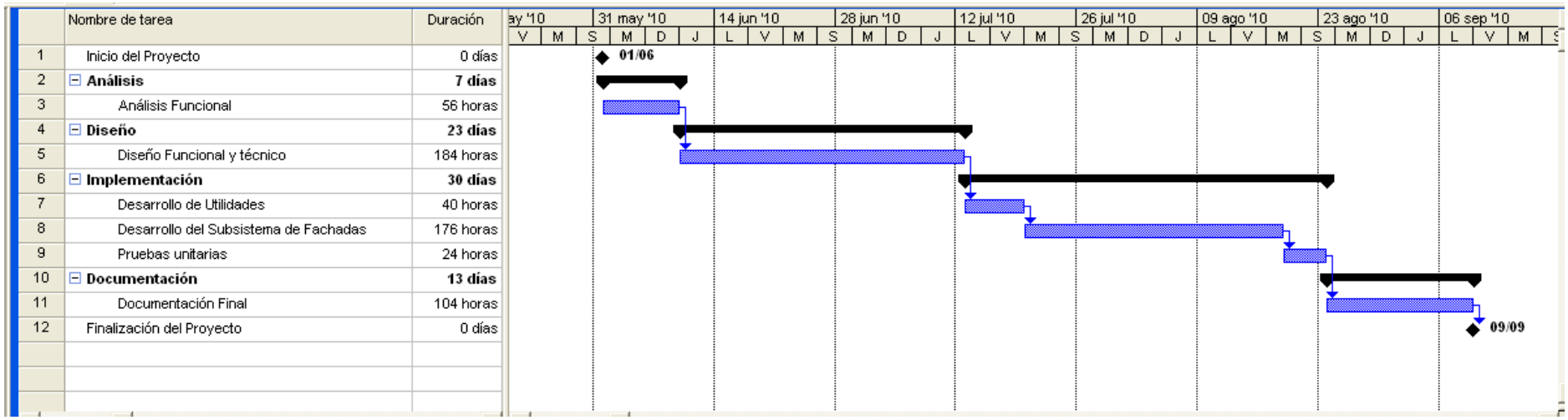

Figura 12.1.1 – Planificación del Proyecto

12.1.2 Coste del personal

Se considerará a las personas necesarias para el desarrollo de este proyecto con el mismo salario y perfil de Analista/Programador.

A todos los costes representados habría que aplicarles el Impuesto de Valor Añadido (IVA), ya que no se ha considerado para realizar los cálculos.

Se pretende obtener el coste por hora por lo que se tendrán en cuenta los siguientes parámetros para realizar el cálculo:

- Días laborables / mes: 20
- Horas trabajadas / día: 8 horas
- Horas trabajadas / mes: 160 horas
- Salario bruto por persona y año: 24.000 €
- Horas en un año = 1.920 horas.
- Coste bruto por persona y hora = Salario bruto por persona y año / horas en un año

Por lo tanto el coste de la hora sería de 12,5 €

En la siguiente tabla mostramos el número de horas imputadas en cada tarea del proyecto y el coste que supone teniendo en cuenta el coste por hora.

Tarea	Nº de Horas	Coste / € (Sin IVA)
Análisis	56	700
Diseño	184	2.300
Implementación	240	3.000
Documentación	100	1.250
TOTAL	580	7.250

Por lo tanto el coste del personal del proyecto sería de 7.250 €.

Dentro de las horas de cada una de las tareas se ha tenido en cuenta la búsqueda, selección, traducción y generación de documentación así como búsqueda de

estrategias y métodos para obtener soluciones, herramientas para análisis y diseño y programas para la implementación de software.

12.1.3 Ejecución de material

Para la realización del proyecto ha sido necesaria la compra de material informático, así como licencias de software y conexión a Internet para consulta de datos.

Descripción	Coste / € (sin IVA)
Compra de equipo informático	600
Compra de impresora láser	70
Licencias de Software	300
Conexión a Internet	140
TOTAL	1.110

12.1.4 Material fungible

A continuación se indican los costes del material consumible necesario.

Descripción	Coste / € (sin IVA)
Material de impresión	140
Material de oficina	50
Encuadernación	220
TOTAL	410

12.1.5 Costes de Instalaciones

La realización del proyecto conlleva el alquiler de un puesto de trabajo durante 4 meses.

En el precio del alquiler están incluidos los costes de suministros y otros servicios, éstos serían:

- recepción de correspondencia y paquetería
- recepción telefónica
- limpieza
- consumo de agua y electricidad
- climatización
- video vigilancia
- utilización de salas de reuniones

Descripción	Coste / € (sin IVA)
Alquiler de puesto de trabajo	140 /mes
TOTAL	560

12.1.6Coste final del proyecto

Con los costes expuestos anteriormente se calcula el coste total del proyecto.

Descripción	Coste / € (sin IVA)
Coste de personal	7.250
Ejecución de material	1.110
Material fungible	410
Instalaciones	560
TOTAL	9.330

12.2 Diccionario de Términos

ADL/SCORM	*Advanced Distributed Learning / Sharable Content Object Reference Model*. ADL es un perfil de aplicación que combina muchas especificaciones y las particulariza para un caso concreto. Con SCORM propone un entorno de ejecución, un modelo de metadatos y un modelo de la estructura de los cursos
AICC	*Aviation Industry Computer-Based-Training Comitee*. Asociación internacional de capacitación de profesionales basada en tecnología
ALL	*Accesible Lifelong Learning*
ATAG	*Web Authoring Tools Accessibility Guidelines*. Directrices de accesibilidad de herramientas de autor que considera el software para construir sitios web con contenidos validos e interfases accesibles
CC/PP	*Composite Capability/Preference Profile*. Perfil de Dispositivo dentro de las características de usuario
CO	Objetos de Contenido (ej: 1 hora de video). Unidad mas pequeña de contenido útil y autónoma
CPSM	Módulo de Software de Personalización de Contenido
D4all	Diseño para todos
Dublin Core Metadata	Especificación nacida con el objeto de describir recursos de carácter genérico en la Web.
e-learning	Aprendizaje electrónico. Forma de estudio autodidacta la cual puedes utilizar para ampliar tus conocimientos del área de tu interés utilizando la red como base de investigación
EHEA	*The European Higher Education Area*. El Espacio Europeo de Educación Superior es un ámbito de organización educativo que quiere armonizar los distintos sistemas educativos europeos.
EVE/A	Entorno Virtual de Enseñanza/Aprendizaje
EU4ALL	*European Unified Approach For Accessible Lifelong Learning*

IEEE	*Institute of Electrical & Electronics Engineers*
IMS	Es un esqueleto de especificaciones que ayuda a definir variados estándares técnicos. El *IMS Global Learning Consortium, Inc*. es el principal promotor y desarrollador de especificaciones abiertas orientadas a *e-learning*
IMS AccLIP	*IMS Accessibility for Learner Information Profile*. Define nuevas estructuras de datos para poder especificar preferencias de accesibilidad teniendo en cuenta características como discapacidades
IMS AccMD	*IMS AccessForAll Meta-Data*
IMS QTI	*IMS Question & Test Interoperability Specification*. Especificación de preguntas y evaluaciones o exámenes
IMS-CP	*IMS Content Packaging Specification*. Describe el modo en que se debe empaquetar el contenido para permitir distribución de contenidos reutilizables e intercambiables
IMS-LIP	*IMS Learner Information Profile*. Indica que información y como debe guardarse referente a alumnos o a un productor de contenido
IMS-MD	*IMS Meta-Data*
ISO IEC JTC1 SC36	*Individualised Adaptability and Accessibility for Learning Education and Training*
LLL	*Life-Long Learning*. Aprendizaje permanente
LMS	*Learning Management System*. Sistema de Gestión de Aprendizaje. Entorno hardware y software diseñado para automatizar y gestionar el desarrollo de actividades formativas, también se conoce como Plataforma de Teleformación. Es uno de los elementos fundamentales de e-learning
LO	*Learning Object*. Objetos de aprendizaje
LOM	*IEEE Learning Object Meta-Data*. Estándar de e-learnig para la definición de Objetos de Aprendizaje.
LTSC	*Learning Tecnology Standards Comitee*

MIT/ OKI/ OCW	*Open Knowledge Initiative, Open Course Ware*. Iniciativa para poner contenidos de cursos en la web gratis colaborando con la iniciativa para crear sistemas de enseñanza abierto e interoperable. Las dos iniciativas pertenecientes al *Massachussets Institute of Technology*
Módulo PC	Módulo de Personalización de Contenido
MOMR	*Media Object Metadata Repository*
UAProf	*User Agent Profile*
Objeto DOM	*Document Object Model*. Modelo de objetos de documento
SOA	*Service Oriented Architecture*
UAAG	*User Agents Accessibility Guidelines*. Directrices de accesibilidad de agentes de usuario que considera los navegadores, los visores multimedia y su interoperabilidad con las diferentes tecnologías.
UNED y UKOU	Las mayores Universidades a Distancia de Europa
VLE	*Virtual Learning Environment*. Entorno de aprendizaje virtual
W3C	*World Wide Web Consortium*
WAI	*Web Accessibility Initiative*
WCAG	*Web Content Accessibility Guidelines*
WWW	*World Wide Web*
XML	*Extensible Markup Language*